68' MICRO JOURNAL

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30-SEK

$2.95USA

VOLUME VIII ISSUE X • Devoted to the 68XX User • October 1986

"Small Computers Doing Big Things"

WE HAVE MOTOROLA COVERED

GESSBS-4 $316*

1 MHz 6809 CPU
Sockets for up to 32 Kbytes EPROM
Sockets for up to 16 Kbytes CMOS RAM
One RS 232 serial port
40 TTL Bidirectional I/O lines
4 x 16-bit timers

GESMPU-14 $636*

8 MHz 68010 CPU
Optional 32081 arithmetic unit
Sockets for up to 128 Kbytes EPROM
One RS 232 serial port
4 x 8-bit timers
Real-time clock/calendar and battery

GESMPU-4A $316*

8 MHz 68000 CPU
Sockets for up to 128 Kbytes EPROM
Sockets for up to 64 Kbytes CMOS RAM
One RS 232 serial port
Three 16-bit timers

SOFTWARE

OS-9®, PDOS®, CP/M 68K®, Editor-Assembler, Basic-Pascal-C compilers, FORTH.

Gespac

USA - CANADA
100 West Hoover Ave.
Mesa, AZ 85202
Tel. (602) 962-5559
Telex 386575

INTERNATIONAL
3,chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

* 100 Piece quantities

Ⓜ is a registered trademark of Motorola Inc.

GMX Micro-20 prices

MICRO 20 (12.5 MHz)	$2565.00
MICRO 20 (16.67 MHz)	$2895.00
8 PORT RS232 BOARD SET (SBC-8S)	$ 498.00
PROTOTYPING BOARD (SBC-WW)	$ 75.00
BACK PANEL PLATE (BPP-PC)	$ 44.00
I/O BUS ADAPTER (SBC-BA)	$ 195.00

QUANTITY DISCOUNTS ARE AVAILABLE ON THE ABOVE ITEMS AS FOLLOWS:4-9,LESS 5%; 10-24,LESS 10%; 25-99,LESS 20%; 100 UP,LESS 30%.

MC68881RC12	$ 295.00
MC68881RC16	$ 395.00
SBC ACCESSORY PACKAGE (M20-AP)	$1690.00

For other configurations and options, contact GMX.

MOTOROLA 68020 USERS MANUAL	$ 18.00
MOTOROLA 68881 USERS MANUAL	$ 18.00

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GMX

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

GMX S-50 BUS prices

68020 SYSTEM

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer, DMA transfers, high speed MMU, we have the UNIFLEX-VM 68020 development system.

The system CPU provides protection to the system and other users from crashes caused by defective user programs.

The system's intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions.

The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. It allows up to 4 Megabytes of Virtual Memory per user. All systems include 1MB of static RAM, one 3-port Intelligent Serial I/O board, DMA Controllers, a 5" 80 track floppy drive.

PRICES

#020 UniFLEX VM with 25MB HD	$10,980.20
#020 UniFLEX VM with 85MB HD	$12,480.20

YOU CAN EXPAND THESE 020 SYSTEMS WITH;

60MB STREAMER	$ 2,400.00
REMOVABLE PACK DRIVE	$ 1,200.00

INTELLIGENT I/OS

#14 3 Port Serial-30 Pin	$	498.14
#13 4 Port Serial-50 Pin	$	618.13
#12 Parallel-50 Pin	$	538.12

CABLE SETS FOR I/OS

# 95 Cable Sets Specify Card	$	24.95
# 51 Cent. B.P. Cable for #12 & #44	$	34.51
# 53 Cent. Cable Set	$	36.53

6809 SYSTEM

The number 39 systems include:#05 CPUwDAT; #19 Classy Chassis; 256K Static RAM; a # 43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates;

System # 39 OS-9 GMX II Dual 80 DSDD		$ 2,998.39
"	w19MB	$ 4,698.39
	w72MB	$ 6,298.39

The Software included in this System: GMXBUG monitor; FLEX; and OS-9 GMXII. You can software select either FLEX or OS-9. Also includes OS-9 Editor, Assembler, Debugger, BASIC-09, RUNB, RMS, DO, and GMX-VDISK for FLEX,

System # 39 UniFLEX	w25MB	$ 4,698.39
"	w85MB	$ 6,298.39

The UniFLEX Operating System is included.

6809 SYSTEMS USING THE GIMIX III CPU & INTELLIGENT I/O PROCESSOR BOARDS

These System include: GMX6809 CPU III; one #11 3 port intelligent serial I/O & Cables; #19 Classy Chassis; 256K Static RAM; #68 DMA controller; all necessary cables, power regulators, and filler plates.

System # 79 OS9 GMX III Dual 80 DSDD		$ 4,498.79
"	w25MB	$ 6,498.79
"	w85MB	$ 7,998.79

The # 79 System Software includes: OS9 GMXIII; OS9 Editor, Assembler, Debugger, BASIC 09, RUNB, RMS, DO, RAMdisk, O-FLEX; GMXBUG; FLEX. The GMX Support ROM and the hardware CRC board are exclusive features included in this system.

System # 89 UniFLEX III	w25MB	$ 6,798.39
	w85MB	$ 8,298.39

The UniFLEX GMX III Operating System is included.

A Member of the CPI Family

68 Micro
6800 6809 68000 68010 68020
Journal

10 Years of Dedication to Motorola Users

Editorial Staff

Publisher:
Don Williams Sr.

Executive Editor:
Larry Williams

Production Manager:
Tom Williams

Administration:
Office Manager:
Mary Robertson
Subscriptions:
Joyce Williams

Contributing & Associate Editors:

Ron Anderson
Ron Voigts
Doug Lurie
David Lewis

Dr. E.M. Bud Pass
Art Weller
Dr. Theo Elbert
& hundreds more of us

Contents

68 MICRO JOURNAL

"Contribute *Nothing* - Expect *Nothing*" DMW 1986

COMPUTER PUBLISHING, INC.
"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
CPI
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 - Telex 510 600-6630

68 Micro Journal is published 12 times a year by Computer Publishing, Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year: $24.50 USA, Canada & Mexico $34.00 *a year. Others add* $12.00 *a year surface, airmail add* $48.00 *a year, USA funds!* 2 Years $42.50, 3 Years $64.50 *plus additional postage, for each* additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number and date, as well as a statement that the material is original and the property of the submitting author. Articles submitted should be on diskette, Macintosh, OS-9 or FLEX format. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please.

Please do not *format with spaces any text indents, chart items, etc. (source listings o.k.) WE will edit in ALL formatting*. Text should be flush left column and use ONLY a carriage return to separate paragraphs or other article text items! MacWrite, FLEX TSC, Stylo formatting acceptable.

Letters & Advertising Copy

Letters to the Editor should be original copy, signed! Letters of gripe as well as praise are acceptable. *We reserve the right to reject any letter to the editor or advertising copy material, for any reason we deem advisable.*

Advertising Rates: Commercial please contact 68 Micro Journal advertising department. Classified ads must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word after the first 15. All classifieds must be pre-paid. No classifieds accepted by telephone.

The VME BUS and OS-9:

Ultimate Software for the Ultimate Bus.

Modularity. Flexibility. High Performance. Future growth. These are probably the prime reasons you chose the VME bus. Why not use the same criteria when selecting your system software? That's why you should take a look at Microware's OS-9/68000 Operating System—it's the perfect match for the VME bus.

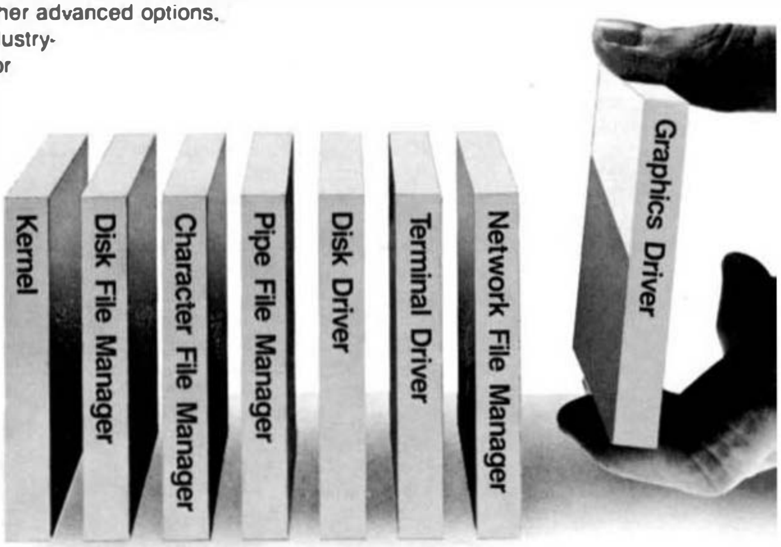
When you're working with VME you must have access to every part of the system. Unlike other operating systems that literally scream KEEP OUT!, OS-9's open architecture invites you to create, adapt, customize and expand. Thanks to its unique modular design, OS-9 naturally fits virtually any system, from simple ROM-based controllers up to large multiuser systems.

And that's just the beginning of the story. OS-9 gives you a complete UNIX-application compatible environment. It is multitasking, real time, and extremely fast. And if you're still not impressed, consider that a complete OS-9 executive and I/O driver package typically fits in less than 24K of RAM or ROM.

Software tools abound for OS-9, including outstanding Microware C, Basic, Fortran, and Pascal compilers. In addition, cross C compilers and cross assemblers are available for VAX systems under Unix or VMS. You can also plug in other advanced options, such as the GSS-DRIVERS™ Virtual Device Interface for industry-standard graphics support, or the OS-9 Network File Manager for high level, hardware-independent networking.

Designed for the most demanding OEM requirements, OS-9's performance and reliability has been proven in an incredible variety of applications. There's nothing like a track record as proof: to date, over 200 OEMs have shipped more than 100,000 OS-9-based systems.

Ask your VME system supplier about OS-9. Or you can install and evaluate OS-9 on your own custom system with a reasonably priced Microware PortPak™. Contact Microware today. We'll send you complete information about OS-9 and a list of quality manufacturers who offer off-the-shelf VME/OS-9 packages.



Modular Hardware Deserves Modular Software

MICROWARE.

Microware Systems Corporation
1866 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

Microware Japan, Ltd.
41-19 Honcho 4-Chome, Funabashi City • Chiba 273,
Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB
S:t Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
Phone: 018-138595
Telex: 76129

Dr. Rudolf Keil, GmbH
Porphystrasse 15
D-6905 Schriesheim
West Germany
Phone: (0 62 03) 67 41
Telex: 465025

Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83-3377
Telex: 828275

Vivaway Ltd.
36-38 John Street
Luton, Bedfordshire, LU1 2JE
United Kingdom
Phone: (0582) 423425
Telex: 825115

Microprocessor Consultants
92 Bynya Road
Palm Beach 2108
NSW Australia
Phone: 02-919-4917

Microdata Soft
97 bis, rue de Colombes
92400 Courbevoie
France
Phone: 1-768-80-80
Telex: 615405

OS-9 is a trademark of Microware and Motorola. PortPak is a trademark of Microware. GSS-Drivers is a trademark of Graphic Software Systems, Inc. VAX and VMS are trademarks of DEC. Unix is a trademark of AT&T.

MUSTANG-020 Super SBC™

DATA-COMP proudly presents the first
Under $5000 "SUPER MICRO".

The MUSTANG-020™

CPI
Mustang-020™
A DATA-COMP
Hi-Speed
Product

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, theMUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP
Installed Systems World-Wide
OVER 10 YEARS OF DEDICATED QUALITY!

CPI
A Division of
Computer Publishing, Inc.
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 810 600-6630

MUSTANG-020 Benchmarks **
Time Seconds

Type System	32 bit Int. Loop	Register Long Loop
IBM AT 7300 Xenix Sys 3	9.7	No Registers
AT&T 7300 UNIX PC 68010	7.2	4.3
DEC VAX 11/780 UNIX Berkley 4.2	3.6	3.2
DEC VAX 11/750 " " "	5.1	3.2
68008 OS9 68K 8 Mhz	18.0	9.0
68000 " " 10 Mhz	6.5	4.0
MUSTANG-020 68020 MC68881 OS9 16 Mhz	2.2	0.88
MUSTANG-020 68020 MC68881 UNIFLEX "	1.8	1.22

```
** Loop:  Main()
          {
            register long i;
            for (i=0; i < 999999; ++i);
          }
```

Estimated MIPS - MUSTANG-020 - 2.5 MIPS
Motorola Specs: Burst up to 7 - 8 MIPS - 16 Mhz

MUSTANG-020™ Software

OS-9

OS-9	$350.00
Basic09	300.00
C Compiler	400.00
Fortran 77	400.00
Microware Pascal	400.00
OmegaSoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Option & Expansion

8 Port expansion RS-232 498.00
(total of 20 serial ports supported)

Expansion for Motorola I/O Channel Modules $195.00

** All Expansion boards:
All expansion boards for old style cabinet will require the 101 expansion cable. Systems ordered with newer PC type cabinets do not require this cable.

101 Expansion Cable $39.95

Sculptor+: We are USA distributors for Sculptor+. Call or write for site or multiple system license & discounts, OEM/Dealer.

Special for complete MUSTANG-020™ system buyers - Sculptor+, $695.00, Save $300.00!

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

MUSTANG-020™

FEATURES

- 12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path processor
- 32-bit wide data and address buses, non-multiplexed
- on chip instruction cache
- object code compatible with all 68XXX family processors
- enhanced instruction set - math co-processor interface
- 68881 math hi-speed floating point co-processor (optional)
  direct extension of full 68020 instruction set
  full support IEEE P754, draft 10.0
  transcendental and other scientific math functions
- 2 Megabyte of SIP RAM (512 x 32 bit organization)
- up to 256K bytes of EPROM (64 x 32 bits)
- 4 Asynchronous serial I/O ports standard
  optional to 20 serial ports
  standard RS-232 interface
  optional network interface
- buffered 8 bit parallel port (1/2 MC68230)
  Centronics type pinout
- expansion connector for additional I/O devices
  16 bit data path
  256 byte address space
  2 interrupt inputs
  clock and control signals
  Motorola I/O Channel Modules
- time of day clock/calendar w/battery backup
- controller for 2, 5 1/4" floppy disk drives
  single or double side, single or double density
  35 to 80 track selectable (48-96 TPI)
- SASI interface
- programmable periodic interrupt generator
  interrupt rate from micro-seconds to seconds
  highly accurate time base (5 PPM)
- 5 bit sense switch, readable by the CPU
- hardware single-step capability
- mounts directly to a standard 5 1/4" disk drive

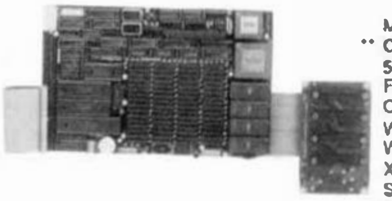
CPI
Mustang-020™
A DATA-COMP
Hi-Speed
Product

Size 8 15/16 x 5 7/8

These *hi-speed* 68020 systems are presently working at NASA, Atomic Energy Commission, other Government Agencies as well as Universities, Business, Labs, and critical applications centers, Worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a ***must***!

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information.

MUSTANG-020 System component prices - Effective July 1, 1986
Prices subject to change - call for latest quotes



MUSTANG-020 (12.50 Mhz)		$2750.00
** Cabinet	(PC or as shown)	$299.95
5"-80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K		$350.00
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Xebec H/D controller		$395.00
Shipping USA UPS		$20.00
Total:		$5059.80

☆☆ <u>DISCOUNT LIMITED TIME</u>: Complete System $1061.00

Complete System $3998.80

WE WILL NOT BE UNDERSOLD!

<u>OPTIONS ADD:</u>

UniFLEX	$90.00	
MC68881 f/p math processor	$275.00	This price subject to increase
16.67 Mhz MC68020	$375.00	Additional MUSTANG systems soon
16.67 Mhz MC68881	$375.00	

Note: Current OS-9 (Ver. 1.2) does not address the MC68881 - Future revisions will. If the 68881 is anticipated in the future, it must be ordered with the system, when originally ordered. UniFLEX does support both the enhanced code of the 68020 and 68881 now.
OPTION BOARDS: ** *Option boards to be installed in Mustang-020 cabinets must be ordered with the extension cable.* The cabinet is too tight for direct plugon. Or specify our new PC type cabinet, with initial order.

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

service

For Over a Decade!
1,200 Titles +

A Division of

CPI

S.E.
Media

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

[illegible]

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add yourown. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

SOFTWARE USER NOTES

BY: Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

OmegaSoft Pascal

Mustang owners take note that OmegaSoft pascal has undergone a considerable upgrade recently. I am a "beta tester" for them, but by the time you read this the new version should be available. Several things have been done to it. First, the compile step (PC) can now be carried right on through to produce the relocatable object file via the -r option. You still have the option of generating a file with the Assembler source code so that you can examine it.

You will recall a couple of columns ago, I was discussing the FFT program, and I supplied a listing for OmegaSoft Pascal. This new version has the constant array feature implemented, and I was able to set up the sine table directly. I managed to get it right on the first try, and it worked just fine.

Later I tried speeding up the FFT program presented last time. You may recall that a 1024 point FFT using REAL arithmetic took 2.6 seconds. I converted to ingeter with a little trick of using a LONGINTEGER for an intermediate result, and the program (after much head scratching I finally read the manual and got it right very quickly) ran 1.75 seconds. Part of any integer version is a divide by 2 through the whole data array for scaling to prevent overflow, since each pass tends to accumulate larger numbers in the array. It occurred to me early, that I could use the OmegaSoft extension, the shift operator of "C" (>>). I tried that rather then the divide, and the answers came out all wrong. It turned out, though the manual is sparse on information, that the shift is unsigned. I therefore got the wrong answer when I shifted the number one place to the right, if the number happened to be negative. I wrote a little procedure to test the number, save the sign, take the absolute value, shift it, and negate if it had been negative, and it worked, but all the overhead lost more time than was gained by not using the straightforward divide by 2.

! Take Note !

Omegasoft Pascal
has undergone a
considerable upgrade...

All this is leading me to the inevitable, getting into Assembler for the 68000, to write my own fast unsigned divide. I do wonder, however, if I will ever get the FFT to run significantly faster than the earlier version with REAL arithmetic, and I know from the tests, that accuracy will suffer because of the "dynamic range" of the signals involved. One is far better off using REAL arithmetic because there, at least, each value is properly "scaled" by the appropriate adjustment of the exponent within the REAL math package. Maximum accuracy is maintained, and little is lost in scaling.

In my applications for the FFT, we are looking at fairly low frequency signals, and it works out that I need to sample the signal for ten seconds anyway. If the user must wait that long (a physical law dictates that) to gather data, it makes little difference if the calculation takes two seconds or three seconds, still, it is a challenge to try to do as well as possible.

A late note is that I have heard again from Bob Reimiller, that the latest update beta test version with double precision REAL arithmetic is about to arrive in my mail. I have promised to do some thorough testing of the accuracy of the scientific functions. I had to report to Bob that *I hadn't yet found any bugs in the last test version that he sent.* He seemed a bit disappointed.

MorePAT

I'm sorry if the columns are beginning to sound like a commercial. I am spending a lot of time on PAT again, and I have a few more additions to tell you about. In the 68K version, I had implemented a little change to make the cursor blink when in INSERT mode. I used the command available on some terminals to change the cursor attribute to blinking. A few days ago I was feeling sorry for myself that one of my terminals has cursor on/off control but not attribute control. You can have a solid cursor or a blinking cursor but the choice is made through dip switches and not through software. I thought a bit about the problem and decided that I could blink the cursor by alternately turning it on and off while sitting in a loop waiting for the user to hit a key. I put a counter in the loop and made a software flip-flop to reverse the state of the cursor every time it the counter runs out. It works fine, and anyone who has a terminal with cursor on/off control can take advantage of it. In addition to that added feature, I have fixed PAT so it will work with terminals that use character positions on the screen to switch video attributes, as well as those that do not.

I also modified PAT so that it will work with terminals that have only a single command string to toggle the cursor on and off. If you have such a terminal you simply put the same code in for cursor on as for cursor off. A variable in PAT now keeps track of the status of the cursor.

Useful Utility

Did you ever get an idea that was so obvious that you wondered why you didn't think of it before? One of those hit me this afternoon as I was thinking about ways of improving our management of our software at the company. Let me give you some background. We use PL/9 for all of our programs. The software we write is for stand-alone hardware in an instrumentation application. The hardware is used to process signals from one or two vibration transducers on a balancing machine to measure the unbalance in a rotating part, and tell the operator what he must do to correct that unbalance. Our machines tend to be "almost alike" but there are always little differences dictated by the customer's parts and the method chosen to add or remove material from the part to balance it. We frequently generate software for a "slightly special" machine by putting together pieces of programs for previously built machines. Some machines don't need some standard feature in others, and if we are tight for program space, we might just remove a procedure that is unused in a particular application.

With all that in mind, you might see how we could at times have a list of variable declarations in which we might declare a few variables that are never used in that particular program. "Why not write a utility that will read the program and make a list of variables, then check to see which ones if any are not found in an assignment statement somewhere in the program?" Of course it immediately gets more complicated than that. Each procedure can have local variables, and the program must check to see that all these are assigned at least once within that procedure, and report any that are not, as excess.

Well, with some excitement, I started blocking it out with REM statements in BASIC, and then filling in the various sections and subroutines. By late evening (I am a one-track, and when I get started on a project like that, it comes home with me, and I work on it until I reach some goal, which in tonight's case was the debugging of the portion that I had written this afternoon. At this point, it will read a PL/9 program and tuck away the names of all the GLOBAL variables and those declared AT specific adresses, in a string array. Having accomplished that, I dummied in a quick loop to list the variable names and assured myself that they were all there. Of course the complications arose. Arrays are declared by name and dimension as in BASIC, and I only wanted the name of the variable in my list. The GLOBAL section of a PL/9 program declares the variable type, and a list of the variables of that type separated by commas. When that list ends and the user wants to change types, he terminates the list with a colon and declares another data type and a list of the variables of that type. The end of the Global variable list is marked with a semicolon. Variables may be declared one to a line or many on a line. I usually put one on a line and follow it with a comment. Of course my program had to skip comments and go on to the next line for more variables. It had to skip dimensions in parentheses also.

Eventually I completed that part of the task. Next comes the smaller loop to check each procedure for local variable use and the making of provision to open and read INCLUDEd files as part of the overall program as well. Presently I am thinking of collecting the global variable list and checking the local variables in one pass through the program file, and checking the global variables on a second pass.

I've done a couple other utility programs previously, and then compiled them in K-BASIC. I call one of them LISTALL, which will do a printed listing of a PL/9 program, and included files complete. There is a simple switch to turn off the listing of any of the INCLUDE files. A comment /*NOL*/ following the include statement will turn off the listing of that particular include file.

The second program is called SKELETON, and it prepares a list of all the procedures in the program and the include files. It indexes each procedure as to its line number in its respective file. Both of these are handy for listing programs and indexing procedures to aid in locating them for debugging of a long program. (The last complete listing I made of PAT, with all the include files is 54 pages).

This last program presently called VAR to keep the name short for frequent save and load operations, will fill out a trio of useful programs for housekeeping a PL/9 program, and with a few fairly simple changes would "housekeep" a Pascal or C program or module just as well. When the present effort is finished, I will publish all three of these programs (might have to go one a month for space reasons). Part of the reason I'd like to dc this is just to let you know that I do still use BASIC for some non-trivial programming.

Stylograph

I recently decided to configure Stylo for my latest terminal acquisition, the Tandy DT-1. I found no mention of it in the list of pre-configured terminals, nor of the Televideo 910, nor any of the other three that it emulates. I found it not too difficult to work my way through the sample terminal configuration files supplied with Stylo and to come up with a configuration file. A couple of things bothered me, however. Stylo does not correctly handle a terminal that uses the same code for cursor on as for cursor off. It gets an extra toggle, for instance, when you go to the menu. I had to put in "no code" for cursor on and off, though the terminal is capable of it. Other than that problem, however, *Stylo works fine on this new but obsolete terminal*. I am, in fact, using it to write this.

Microware

I owe the folks at Microware an apology for thinking bad things about them. See my remarks in last month's column. The problem that I mentioned with OS-9 adding a CR to my LF turned out to be a matter of how "C" was implemented by Microware. I had used the C library function putc() (for put character to the terminal). It turns out that in "C" that is considered a high level function, and it goes through an OS-9 call that always adds a CR to LF. David West of Microware indicated that I should instead use a lower level function called write() which would output only what I told it to output. I made the switch, requiring only a few lines of code to be changed, and the problem was cured. Mr. West pointed out that although K&R is THE standard for "C" implementations, it is not really a complete specification in every detail, and there are a few ambiguities in it.

I might add that I am most definitely not an expert "C" programmer at this point. As I said in my reply to Microware, a real "C" guru would look at my "C" version of PAT and say that I am a Pascal programmer, just incidentally using "C" to code my program. I practically get lost thinking about the simple argv parameter so commonly used in "C". argv is an array of pointers to arrays of characters. argv[1] is a pointer to an array of characters containing the first parameter that the user put on the command line after the name of the "C" program. Actually, argv[0] is a pointer to an array of characters containing the name of the program from the command line... Anyway, though I think I am finally catching the spirit of "C", I am my no means an expert programmer in it, though as I said to Microware, my PAT in "C" is over 50 pages of program. I tend to use the more familiar array construct in "C" rather than the pointer concept.

I must convey my thanks here to Microware for a very quick and thorough answer to my letter, and a simple solution to my problem. I had also asked another question about a warning message when I compiled PAT. That turned out to be an extremely dumb "pilot error". When it was explained to me, I groaned. I certainly knew better, but the real reason for the warning never dawned on me. The answers given to me would doubtless hold true in the 6809 versions of OS-9 and Microware "C" as well.

In his letter Mr. West referred to an article he had written in "Pipelines", which I assume is a publication of Microware with which I am not familiar. If Microware has made any mistake at all, it is simply the assumption that customers are already somewhat familiar with OS-9. Perhaps they simply assumed that all purchasers of OS-9/68K had owned or used OS-9/6809, and were therefore familiar with it. In my case, that was a bad assumption.

I indicated in my reply to Mr. West's letter, that the documentation is well organized and just about complete, but not abundantly clear to a novice OS-9 user. For example, the documentation on OS-9 refers to RBF devices and SCF devices when discussing device descriptors. It is more or less inferred after a lot of reading that the SCF devices are terminals and the RBF devices are mass storage devices such as floppy or hard disk drives. I have not yet come across a defmition of either of these or an explanation of the mnemonic RBF or SCF. Neither the glossary nor the index of the user's manual have entries for these terms. I guess the terms are so familiar to the folks at Microware that they don't realize that a new user of OS-9 doesn't have the foggiest notion what they are talking about. It is the old problem of the manual writer being too familiar with the product.

Let me hasten to add, however, that aside from a few undefined and initially cryptic terms, the Microware manuals are done in a most professional manner. Generally, I have been able to find answers to all my problems by referring to them, and I have been able to locate the need information quickly, indicating that the information is well organized. The manuals are typeset and not Xerox copies of dot matrix printed text. They are supplied in the "PC standard" size three ring binders. When I received an update on Microware "C" for the system, it included a complete new manual, in contrast to Lattice "C" for the IBM compatibles which was initially supplied with a manual that was about three upgrades obsolete, and a stack of addenda and correction sheets!

I've lately become very sympathetic, or should I say empathetic with software suppliers with regard to upgrades. *No software product of any complexity will be bug free when it is first introduced on the market.* Bug repairs sometimes lead to requirements for changing the documentation. Any programmer or team worth anything at all, will see opportunities for improvements in the way of code simplification, user interface simplification, and new features, and they will want to include these new featues in later releases of the product. All of these tend to create the need for documentation changes. Reading a 50 page instruction manual for the 123rd time while paying enough attention to catch the things that have to be changed is no little chore. Having made the changes, it is then necessary to stamp out all the old copies of both the documentation and the software, a chore that is even harder if you are carefull to keep backups and backup backups.

Trees

I note with interest, the article by Mickey Ferguson in the latest '68' about HIER, the software that adds utilities and a patch to FLEX to allow true tree structured directories. I have been working on a similar batch of programs for myself, but had not gotten to the point of modifying FLEX yet. I guess I will take the easy way out and order HIER. I was glad to learn that the approach was not simply that of splitting the hard disk into "volumes" which in my opinion can only be a massive space wasting technique. In looking at my directories on our MS-DOS machine and the Mustang with OS-9, it is obvious that some of the directories use a very large number of sectors, and others use few. Dividing a disk up into, say 10 volumes would probably waste at least half the available space, at least for me in my applications.

ASSEMBLER

I have a problem of no small proportion. My son wants to learn how to program those "neat game programs" on the Atari computer. I realize that he doesn't have any idea of the level of programming expertise that is required, nor of the time it will take to write a single game program with graphics and action, but I thought at least I would start him on the road of understanding what is inside of a computer and how a program is written in Assembler. My problem is basically, where do I start? I started by buying a single board computer and reading the manuals, but I was considerably older at the time than my son is now, and I had the advantage of a degree in Electrical Engineering and several years of experience.

My son, David has done some programming in BASIC, and has taken a couple of "community" offered courses outside of school, so he has some idea of the process involved in writing a program, but he has no idea of what goes on inside the computer. Do we start with the architecture of the processor? I have already decided to start with the 6809 since I have all the necessary software for editing and debugging an assembler program. Once the processor is understood, is the next step a discussion of a typical memory map? Do we then talk about the monitor, bootstrap loader, and disk operating system? Maybe with that all understood we can begin to program something simple. Or will the average curious person have lost all interest by the time he has been through all that "theory" without having written and run a simple program?

Perhaps a better approach is a quick explanation of the processor architecture followed by a quick explanation of a memory map, a mention of a monitor program and how it is used to modify memory and run a program. Then, having actually added 2 and 2 and seen the result 4, to go back and fill in more about the processor and the monitor. Then, discuss the disk operating system and the routines that are built into the system, such as the calls to print a string or a number to the terminal. I strongly suspect the latter approach will be more interesting for a longer time. I will keep you informed of our progress and perhaps use what I learn about teaching, as the basis for a short course here in this column.

More FFT

I made the mistake, in the August column of mentioning that I had found an error in the Uhrich algorithm for the FFT without specifically saying what it was. I guess it was a case of the author being too familiar with the subject again... Anyway, I should have spelled out the error and the fix. The problem was that the second part of the complex multiply was omitted. The two lines that calculate M.re and M.im were:

```
M.re := x[1,c].re * cosw;
  M.im := x[1,c].im * sinw;
```

The correct version is:

```
M.re := x[1,c].re * cosw + x[1,c].im * sinw;
  M.IM := x[1,c].im * cosw - x[1,c].re * sinw;
```

The corrections were made in both my Pascal program listing and the PL/9 program listing published in the August column. To clarify my remarks in that column. The program with the error ran fast but gave the wrong answers. With the correction, the program runs about 40% longer, but the answers are correct. Even with the additions, this program outperforms the Cooley - Tukey algorithm considerably. I apologize for not giving the corrections specifically when I mentioned the error.

OmegaSoft Pascal

I just received the latest pre-release copy of the next version of OS-9 /68K Pascal from OmegaSoft. It now supports LONGREAL data type, which, of course is a 64 bit floating point arithmetic type. I checked the scientific functions and found them all to be accurate to 15 digits and better. One thing I like about this package as opposed to the "C" packages with "DOUBLE" floats, is that when a "C" package has this capability, all floating point calculations are carried out in double precision, and then half of the precision thrown away if the result is specified as float rather than double. That is, you can't gain speed by using the shorter REAL data type when the 7 digit precision will do. (Unfortunately, this is the mode of operation described by K&R in their book, the default standard for "C" compilers). OmegaSoft Pascal is a different story. If you use REAL data types, you get 6 or 7 digit precision. *If you use LONGREAL, you get 15 or 16 digits*! The addition of LONGREALs didn't change the benchmark results with the REAL arithmetic, that I reported some time ago here. I have not had a chance to rework the benchmark to use LONGREALS and see what time penalty is paid for the extra precision. Theoretically, at least, it should take four times as long to multiply to double the precision

CAD Again

I mentioned the SmArtwork package from Wintek for PC board layouts, in the August column and indicated that we were very happy with it. Unfortunately, I have to say that at least for the present I can't say the same for their "schematic capture" software called Hi Wire. We recently sent off for two Demo packages and two "money back trial" packages for the Tandy 3000. Of the four packages, three of our people rated Hi Wire last. It was certainly the most capable, but it had several shortcomings. First, it took more keystrokes than other CAD packages did to do the "graphics editing" functions. Secondly it had no coordinate system of any kind. You had no way of knowing where you were on a sheet of paper. Thirdly, it was very easy to pan off the edge of what you had done using the mouse, and have no idea whatever of where the drawing was. We didn't like Engineering Designer or ORCAD much better. We decided the money would be better spent in purchasing the 80287 for the Tandy, and in upgrading our present AutoCad package.

Late News

I've received the release version of the OmegaSoft Pascal mentioned above, but have not had time to get it fired up. I assume it is quite like the last test version I received.

Please note that I am not knocking Wintek. We still think SmArtwork is GREAT, even more so since we had a chance to look at the artwork generated by someone else's package. There is a lot of potential in the Hi Wire package, but we didn't think it was yet far enough along to be a salable product. If Wintek continues to enhance it as they have SmArtwork, it will be very nice in the future sometime.

EOF

C User Notes™

By: E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants
1454 Latta Lane, N. W.
Conyers, GA 30207
404-483-1717/4570

INTRODUCTION

This chapter continues the discussion of the proposed ANSI C standard and the discussion of common problem areas in the use of the C language and its libraries. It also provides a few puns based upon the C syntax.

PROPOSED ANSI C STANDARD

The scope of the proposed ANSI C standard includes the following:

the representation of C programs,

the syntax of the C language,

the semantic rules for interpreting C programs,

the representation of input data to be processed by C programs,

the representation of output data produced by C programs,

the restrictions and limits imposed by a conforming implementation of C.

The scope of the proposed ANSI C standard excludes the following:

the mechanism by which C programs are transformed for use by a data processing system,

the mechanism by which C programs are invoked for use by a data processing system,

the mechanism by which input data are transfonned for use by a C program,

the mechanism by which output data are transformed after being produced by a C program,

the size or complexity of a program and its data that will exceed the capacity of any specific data processing system or the capacity of a particular processor,

the minimal requirements of a data processing system that is capable of supporting a conforming implementation.

Non-extern identifiers may be up to 31 significant characters in length, in which no case or character folding is allowed. The standard recommends, but does not specify, similar interpretation for extern identifiers. The problem with extern identifier length and contents is that system link editors often apply arbitrary length and content rules to external symbols. These may be as restrictive as six to eight upper-case letters and/or digits, starting with a letter.

Data types have been somewhat expanded in scope and tightened in usage.

Data types const, enum, signed, void, and volatile have been added. Keyword entry, which was never really defined, has been dropped. Data type long float has been dropped in favor of double. Data type long double is allowed as an implementation-dependent feature.

Int and long data types are signed by default, but may be denoted as unsigned. The definition of unsigned or signed char is left to the implementor of the conforming C compiler.

Const data types denote values which will not change for the duration of the execution of the program. They were placed into the standard to assist compiler writers in optimizing the machine code produced by conforming C compilers. Expressions involving only constants are automatically provided with const data type; however, expressions involving constants and const data types are not automatically provided with const data type. The programmer is prohibited from changing the value of a const data type, in a manner analogous to the restrictions on changing the base location of an array.

Volatile data types denote locations which may behave in unpredictable manners. They were placed into the standard to provide the programmer a method of disabling optimization of special code or locations. An example of an application of a volatile data type is for memory-mapped locations which may appear to contain different values when fetched than when stored. An optimizing compiler might note that a location has apparently not been changed between a store operation and a fetch operation and eliminate one or both operations. The volatile data type is intended to allow the selective

Void data types have several uses in the new standard. A void function is assumed to return no value, rather than the standard int value. This may assist in the optimization and verification of C functions. A void function parameter in a prototype (discussed later) indicates that no parameter is required or allowed. A pointer may be coerced by (void *) to indicate that the pointer is to have no alignment restrictions applied to it in that particular expression.

Enum data types provide a simple means of enumerating sets of ints. The identifiers in enum lists are declared as constants with type const signed int and value normally equal to their base-zero location in the list, although this may be overridden. An enum data type is used in a similar manner to a struct or union data type. However, the enum'ed variables may be used as if they had data type of const signed int.

The identifiers used to name struct or union members are placed into a separate name space from the struct or union names and from functions and other data types, as are the labels used in goto statements. Structure and union members are tightly bound to ıeir structures and unions and may not be used outside these bounds, whether as if they were other data types or as if they were declared as part of other structures or unions.

Structure and union elements, not only pointers to structure and union elements, may be passed to functions, and used in expressions. When they are passed to functions, a copy, not the original element, is passed, so the actions are similar to the handling of other data types.

C PROBLEM AREAS

String handling in C is simultaneously one of the weaker and one of the stronger points of the language. It is one of the weaker points in that the C language supports only very primitive usage of strings, not supporting string assignment or comparison, which are provided in many other languages such as BASIC. It is one of the stronger points in that the C language does not restrict the programmer in terms of defining operations on strings, although perhaps at the expense of elegance of expression.

In particular, many BASIC language processors assume some small maximum length (like 255) and/or terminating character for strings, significantly reducing the generality of the string processing capabilities in that processor. Many use quite crude methods for allocating and deallocating string space, often causing a severe performace degradation when performing large amounts of string processing.

By defining and using sophisticated string-processing libraries, the C user may avoid many of the difficulties associated with the lack of string handling in the C language.

The user is responsible for ensuring that the strings are placed into the desired areas. The C language provides no automatic memory protection, as is provided by BASIC and some other languages. Providing a subscript to an array that is too large or negative is all too easy and often leads to a system crash or some unrelated and misleading error message due to the corruption of some other area of memory.

The standard C library contains several traps for the unwary user. Return codes and order of parameters are inconsistent, even for similar functions. Since strings are assumed zero-delimited by most of the standard functions, there is no automatic protection against too-long strings. The experienced C user develops several defenses against some of these problems.

When there are several functions which accomplish almost the same function, use the ones which are safer, more basic, or more familiar.

The use of the gets(string) function is unsafe since the length of the input data is undefined until actually read, whereas fgets(string,count,fileptr) is safer since the number of characters which will be placed into string is limited.

The putc(chr,fileptr) function has its arguments reversed from fprint(fileptr,...) and similar functions. Also, the putc function normally returns its first argument, but if a character code which causes putc to return -1 is input to the function, a false error condition will be encountered.

The memory allocation functions calloc(count,size), malloc(count), and realloc(ptr,count) all return NULL upon failure to allocate, but brk(ptr) and sbrk(count) both return -1 upon failure (and sbrk has data type char *).

The string movement function strcpy(dest,source) and similar functions assume zero-delimited strings and are unsafe in not limiting the number of characters moved. The function strncpy(dest,source,maxcount) is safer in that the number of characters moved is limited to maxcount. However, the destination string will not be zero-delimited if maxcount characters are moved.

C PROBLEM

Following are the two short C functions with errors which were shown in the previous chapter.

This function writes garbage characters to a file, rather than the values of the characters passed to it.

```
int outchar(fd, ch)
int fd;
char ch;
{
   return write(fd, ch, 1);
}
```

The problems is in the return statement. The write function requires the address of the data to be written, not the data itself. Following is a corrected version of the function:

```
int outchar(fd, ch)
int fd;
charch;
{
   return write(fd, &ch, 1);
}
```

This function works on some compilers and not on others.

```
int itexists(filename) /* return 1 if file exists */
char *filename;        /* otherwise return 0     */
{
   FILE *fp;
   int i;

   if (i = ((fp = fopen(filename, "r")) != NULL))
      fclose(fp);
   return i;
}
```

The problems is in the declaration or non-declaration of the type of the fopen function. Although it should be of type pointer to FILE, most C compilers assume that it is of type int, unless told otherwise in an appropriate type declaration. Many compilers provide such a type declaration in stdio.h, but some leave its correct definition up to the programmer. The distinction is important only if pointers and ints have different formats in a specific implementation. Following is a corrected version of the function:

```
int itexists(filename) /* return 1 if file exists */
char *filename;        /* otherwise return 0     */
{
   FILE *fp, *fopen;
   int i;
   if (i = ((fp = fopen(filename, "r")) != NULL))
      fclose(fp);
   return i;
}
```

The expression "x+++++x" could be parsed correctly if it were coded as "x++ + ++x" or as "(x++) + (++x)", although it is confusing to humans and to compilers, and few of either can interpret it without spaces or parentheses. This situation is described in the proposed ANSI C standard as an example of an expression which is not processed correctly with a left-to-right parsing of operators of equal precedence unless spaces or parentheses are used.

Following are two short C functions which malfunction. Explain what is wrong with them and fix them.

This function seems to work correctly itself, but sometimes causes the system to crash after the function has been called.

```
#include <stdio.h>

char *password;
```

```
main()
{
   :
   :
   if (getpasswd(password))
   :
   :
}

int getpasswd(password)
char *password;
{
   printf("Enter password: ");
   return getline(password, 16);
}
```

This function fornats a string for subsequent use, but the string usually contains garbage, rather than the correct contents.

```
main()
{
   char *form();
   :
   :
   printf("%s\n", form(100));
   :
   :
}
char *form(value)
int value;
{
   char string[256];

   sprintf(string, "The value is %d", value);
   return string;
}
```

C HUMOR

The following was collected from several sources. They are all puns based upon the syntax of the C language. Readers are encouraged to send original C puns.

Humor at the expense of K & R:

```
auto bahn;
auto hypnosis;
auto mation;
auto mobile;
auto pilot;

case load:
case of_beer:
case tractor:
case worker:

char ade;
char broiled;
char coal;
charo;
char package;
char woman;

double bubble_bubblegum;
double bunk;
double date;
douple deal;
douple dip;
double double_toil_and_trouble;
double dribble;
double header;
double helping;
double mint;
double or_nothing;
double serve;
double team;
double trouble;
double up;

entry prohibited;
entry way;

float a_loan;
float on_air;
float or_sink;

for (crying(); out(); loud());

goto jail;
goto work;

if (you_leave_me_now);
if (pigs_could_fly);

long dress;
long drink;
long fellow;
long johns;
long jump;
long over_due;
long reach;
long receiver;
long trade;
long underwear;
long walk;
long wait;

main(line)
main(street)

register to_vote;
```

```
return for_deposit;
return to_sender;
return trip;

short break;
short changed;
short circuit;
short pajamas;
short pier;
short sheet;
short skirt;
short stop;
short story;
short straw;
short stuff;
short trip;

sizeof (national_debt)
sizeof (shoes)
sizeof (waist)

static discharge;
static electricity;

switch (on)
switch (engine)
switch (places)

union boss;
union jack;
union label;
union pacific_railroad;
union shop;
union suit;

unsigned check;
unsigned original;
unsigned short story;

void where_prohibited;
```

EXAMPLE C PROGRAM

Following is this month's example C program; it helps investigate the problem of the comparison of unsigned long variables on various C compilers. The problem is that many compilers do not handle unsigned long variables properly. This program assumes a 4-byte two's-complement representation for long variables.

In the case of such signed long variables, the smallest representable number is 0x80000000 and the largest representable number is 0x7fffffff.

In the case of such unsigned long variables, the smallest representable number is 0x00000000 and the largest representable number is 0xffffffff.

```
#include <stdio.h>

main()
{
   char string[128];
   long a, b;

   printf("Enter pairs of hex numbers:\n");
   printf("at least try 0-ffffffff\n");
   printf("   and 80000000-7fffffff!\n");
   while (fgets(string, 128, stdin))
   {
      sscanf(string, "%lx %lx", &a, &b);
      printf("%08lx %08lx s=%c u=%c m=%c\n",
          a, b, sslcmp(a, b),
         uslcmp(a, b), mslcmp(a, b));
   }
   pflinit();
}

sslcmp(a, b)
long a, b;
{
   long c;
   c = a - b;
   return ((c < 0) ? '<' : (c > 0) ? '>' : '=');
}

uslcmp(a, b)
long a, b;
{
   long c;

   c = (((unsigned long)a) - ((unsigned long)b));
   return ((c < 0) ? '<' : (c > 0) ? '>' : '=');
}

mslcmp(a, b)
long a, b;
{
   long c;

   if (!(c = ((a >> 16) & 0xffffL) -
      ((b >> 16) & 0xffffL)))
      c = (a & 0xffffL) - (b & 0xffffL);
   return ((c < 0) ? '<' : (c > 0) ? '>' : '=');
}
```

- - -

Basically OS-9 ™

By:
Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

BACK TO BASICS

I thought it would be fun to go back to the beginning. We are going to go to a place where many of us started. This month we are getting back to basics, *BASIC* language that is!

Not everyone started with BASIC. But I find that most people involved with computers had their early experiences in some type of BASIC. Some took it as a first course in computer programming. Others got it when they bought a personnel computer. I learned originally to program in BASIC. The lab at which I work used BASIC in its early days for data aquisition and analysis. So, it is not unreasonable to say that many people had early beginnings in BASIC.

There are a lot of pluses for BASIC. Its string handling ability is excellent. Manipulating character data is a snap. BASIC contains a full math handling library. Formatted data can be printed using the PRINT USING function. Input and Output to various devices can done in an easy, straight forward manor. It is an easy to learn. Generally speaking, it can be a nice language to work in.

So, why have many left it? There can be many reasons. Here are a few major ones. First, it is an interpreted language. Program lines are executed by the BASIC interpreter as they are encountered. On the other hand, programs written and compiled to machine code with languages like C or Pascal are usually smaller and definitely faster. Second, BASIC promotes sloppy programming practices. Indiscriminate use of GOTO's can create programs that are hard to follow and debug.

> BASIC has survived !!
>
> BASIC's have become
> more sophisticated.
>
> They have eliminated
> some of the problems
> by offering programming
> alternatives...
> commands that permit
> structured programming...
> and designed to run faster.

Yet, BASIC has survived. The newer PC's are still supplying it. But the BASIC's have become more sophisticated. They have eliminated some of the problems by offering programming alternatives. They now include commands that permit structured programming. Many of them are designed to run faster. They still retain the spirit of BASIC. This month we are going to look at a BASIC compiler. It is ***KBASIC*** from *S.E. MEDIA*..

K-BASIC is a BASIC language compiler. It allows you to write BASIC programs and compile them to machine code. It generates token files which are converted to assembly language code. The code is then assembled to native machine code. This is all accomplished with three programs. There is ***KS***, which creates the token files. Next is ***KO***, to create the assembly language code. Finally, there is ***KA***, the assembler.

Programs are written with your favorite editor. They can be written in a strict BASIC style. Or you can use structured programming techniques. It supports structures like *WHILE END WHILE REPEAT UNTIL LOOP END LOOP & EXIT IF END EXIT*. There are still the old stand-by's, like *FOR NEXT THEN IF ELSE & GOSUB RETURN*. And if you feel it necessary, you can still use *GOTO*.

Line numbers are another option. They can be used, but are not necessary. In fact it is more appropriate to refer to them as labels. Up to 16 characters can be used for a label. The only restriction is that labels must begin in the 1st column of the source code. All other thing must start in column 2 or more. The following code in classic BASIC style is written:

```
10 REM PRINT 1 THRU 10
20 FOR I=1 TO 10
30 PRINT I
40 NEXT I
```

This can be rewritten in a structured style:

```
COUNTER REM PRINT 1 THRU 10
   FOR I=1 TO 10
      PRINT I
   NEXT I
```

Either method will compile by K-BASIC and run about the same. I think the second one is easier to read. Also, lower case can be used. So, use it if you like. Many programmers like to put Keywords in uppercase letters and labels and variables in lowercase.

6 variable types are recognized by K-BASIC. They are created by a variable name and an appropriate suffix appended to it. The following table defines them.

SUFFIX	TYPE	SIZE
none	REAL	9 bytes
$	STRING	1+length
'	BYTE	1 byte
%	WORD	2 bytes
#	LONG WORD	4 bytes
~	DOUBLE LONG	8 bytes

Now, there are 8 sub-divisions of variables. They include non-dimensioned; single and doubled dimensioned; AT for the non-, single and double dimensioned; and virtual single and virtual double dimensioned. AT assigns variables to fixed locations in memory. Although OS9 assigns data to memory at run time, there are instances where items may be fixed in memory. For example, the direct page variables are fixed. The virtual dimensioning lets the program use random files in disk space for storage. A nice feature! It frees up memory for the program.

K-BASIC supports most math functions for real numbers. It has a random number generator. It supports standard string and number functions. There are a few extras that I have never seen in a BASIC before. One of them is *ROM$*. It returns a Roman numeral of a integer that is passed to it. A line in you program like:

```
PRINT ROM$(1986)
will print
MCMLXXXVI
```

There is also psuedo strings *DATE$* and *TIME$*. They return the date and time. *PRINT USING* is available for a formatted output. There are even provisions to include assembly code in the K-BASIC source code.

It handles errors like any good basic should. With an *ON ERROR GOTO Label* all errors will cause the program's execution to go to your error handling routines located at whatever label you specify. *ERR* and ERL will return the error number and line number where it occurred. The error can be corrected. With a simple*RESUME*, control can be returned to program.

There are also compiler directives that can be used to control the final program. *STRING=* will assign a maximum length for strings. *DIGITS=* will set the length that floating point numbers are reported. *PRECISION=* will set the number of digits a floating point number's mantissa. You are allowed *3 to 99* digits of precision.*TRACE* mode can be turned on to follow a programs execution. *LIB* allows routines from files to be included in your program. This names a few of the things that are under you control.

At compile time, other options are available. Labels and variables can be listed. They can be cross referenced. Listings can be paginated. Source code can be added to output code as comments. And there are a few others like compiling without assembling.

I am telling you about all this because future columns will from time to time include K-BASIC programs. Using K-BASIC can help overcome some of the pitfalls of BASIC. *It compiles to machine code*. So programs will operate fast. There is no Run Time library. Everything is self contained in the program. A structured approach to programming can be taken. Line numbers and labels can be used that are meaningful. Yet, there is much that I didn't cover this month. As time goes on, I'll try to tell you more about it.

A LITTLE NUMERICAL ANALYSIS

I tried to decide what I should do this month. At first I considered a bench mark to demonstrate K-BASIC's speed and code size. Somewhere in a back issue of *68 Micro Journal*, this was already done. I decided to wait on this. The other thought was to see if it really works on standard basic programs. I dug around through some old disk files and came up with the program in Listing 1.

It is a program I wrote for a numerical methods class, some many years back, as you can see from the date. The program will take a number of points that fall on a cartesion coordiate system (that's the X-Y plane) and interpolate points between them. This program will also generate an equation for the input. The program will prompt for input. The equation it generates may need a little simplification, but it should work. I named the program *NEWTONS*, after Isaac Newton, who invented the method. The program appeared to be a good candidate for a trial run. I transfered the program to an OS-9 disk and entered:

```
KS NEWTONS
```

After a minute, it reported that there were 4 errors. One was the use of the command CLS which is used to clear the screen and the others were I had used NEXT statements without indicating the index. The statement:

```
FOR I=1 TO 10
NEXT
```

will work in some BASICs. The missing variable is understood. But strictly speaking, it should be there. I changed the CLS to PRINT CHR$(12) and added the missing variables. The second time around, it compiled and ran fine.

The second listing is the same program. I decided to rewrite it. K-BASIC allows a more structured approach to programming. Notice all the line numbers are gone. I used more meaningful labels. Instead of a GOSUB 1200, the main program uses GOSUB XYInput. The other labels are ComputeY, MakeDeltas, MakePolys and Equation. To make things more readable after the label, I added a comment. K-BASIC recognizes the ' and *REM* as comment lines. If a * appears in column 1, the entire line is a comment.

All the familiar BASIC constructions are here. There are FOR...NEXT loops and lots of PRINT and INPUT statements. I also used a LOOP ENDLOOP and a EXITIF...THEN ENDEXIT. These are the same as in BASIC09. Transforming a program from BASIC09 to K-BASIC or the other way should be relatively easy to do.

The variable names are kept the same as in listing 1, but they could be made more meaningful, too! The % after the some of the variables indicate integers. Using integers is far more efficient. In listing 1, floating point numbers are used for indices. Using a floating point number where an integer will do, causes the loop to run perhaps 4 times longer, since floating point math requires more complicated routines. Using integers speeds up the math and the program speed.

Most of the program can still be recognized a BASIC. One line that is a little different is

dir /r/kbasic, /r/ks.run

This line tells the compiler to where is can store and manipulate its working files, and where to find its run time files. In my declaration, /r/kbasic is its working directory and /r/ks.run is its run time directory. By the way, the /R is the device name for my RAM disk.

There is one thing to be aware of when running K-BASIC. It is a compiler. Three seperate stages are invoked when a program is compiled. Temporary files must be written. The process takes time. I run with a RAM disk which helps. Writing programs with another BASIC like *BASIC09 or Kansas City BASIC* can help speed up the debugging process. Once things work well, convert the program to K-BASIC. Also, you will need atleast two disk drives or a double sided drive. Again a RAM disk works great. The manual is thorough covering most everything. The level I version comes with a 140 page manual packed with necessary information.

In the future I will run K-BASIC programs from time to time. I'll try to comment on them to help with the language. I will continue to cover the other fine languages like BASIC09, PASCAL, C Language and assembler code. So, keep reading **BASICALLY** 0S9. Until next time, have a good month.

RN

Editor's Note: KBASIC is also available for FLEX™ from S.E. MEDIA. No 68000 versions have been released.
DMW

LISTING 1

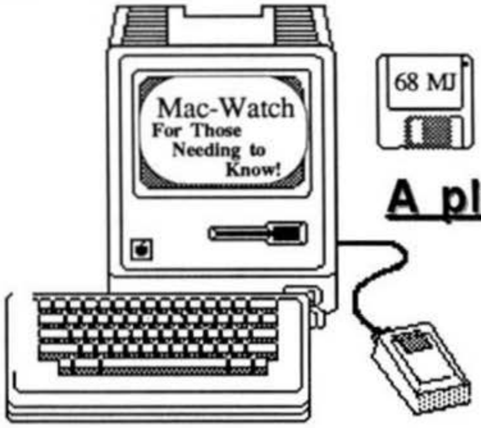
```
 dir /r/kbasic, /r/KS.RUN
1 REM ** NEWTONS **
2 REM      BY
3 REM RONALD D. VOIGTS
4 REM    22 MAY 82
10 REM THE FOLLOWING PROGRAM
20 REM WILL CREATE POINTS
30 REM FOR A GRAPH BASED ON
40 REM DATA PUT IN.
100 PRINT CHR$(12): PRINT
110 PRINT "THIS PROGRAM WILL GENERATE"
115 PRINT "Y VALUES FOR ANY X. A NUMBER"
```

```
120 PRINT "OF POINTS FROM THE CURVE MUST"
125 PRINT "BE ENTERED.  ORDER AND SPACING"
130 PRINT "ARE NOT CRITICAL":PRINT
200 PRINT "ENTER THE NUMBER OF DATA POINTS:"
210 INPUT N
220 N=N-1
300 DIM X(N),Y(N,N)
320 GOSUB 1200
330 GOSUB  1600
340 PRINT: PRINT "DO YOU WISH EQUATIONS TO"
345 PRINT "GENERATE A POLYNOMIAL?"
350 INPUT Y$
360 IF LEFT$(Y$,1)="Y" THEN GOSUB 2200
370 PRINT: PRINT "YOU CAN NOW FIND Y VALUE'S"
499  ••••••••••••••••••••••••••
500 REM COMPUTE Y'S FOR THE X
510 PRINT "ENTER AN X VALUE:"
520 INPUT X
530 GOSUB 1400
540 PRINT "THE Y VALUE IS ....";F
550 PRINT
560 GOTO 510
 1199  ••••••••••••••••••••••••••
1200 REM INPUT X AND Y VALUES
1210 FOR I=0 TO N
1220 PRINT "ENTER X";I;", Y";I
1230 INPUT X(I), Y(I,0)
1240 NEXT I
1250 RETURN
 1299  ••••••••••••••••••••••••••
1400 REM COMPUTE THE Y VALUE
1410 F=Y(0,0)
1420 FOR J=1 TO N
1430 GOSUB 2000
1440 F=Y(0,J)*S+F
1450 NEXT J
1460 RETURN
 1599  ••••••••••••••••••••••••
1600 REM CREATE DELTA'S
1610 FOR J=1 TO N
1620 FOR I=0 TO N-J
1630  Y(I,J)=(Y(I+1,J-1)-Y(I,J-1))/(X(I+J)-X(I))
1640 NEXT I
1650 NEXT J
1660 RETURN
 1699  ••••••••••••••••••••••••
2000 REM CREATE POLYNOMIALS
2010 S=1
20 20 FOR I=0 TO J-1
2030 S=(X-X(I))*S
2040 NEXT I
```

```
2050 RETURN
2200 REM GENERATE POLYNOMIAL
2205 PRINT: PRINT "PRESS <ENTER> TO CONTINUE":PRINT
2210 PRINT "F(X)="
2215 PRINT Y(0,0);"+"
2220 FOR I=1 TO N
2223 IF Y(0,I)=0 THEN 2270
2225 PRINT "(";Y(0,I);")";
2230 FOR J=0 TO I-1
2240 PRINT "(X-";STR$(X(J));")";
2250 NEXT J
2260 PRINT "+"
2265 LINE INPUT "";Y$
2270 NEXT I
2280 RETURN
```

LISTING 2

```
'
' ** NEWTONS **
'
' By: Ron Voigts
' Date:   22 MAY 86
' Language: K-Basic
'
' To compile: ks newtons
'
' The following program will create points
' for a graph based on the input of a set
' of X and Y coordinates.
'
' It can also generate an equation.
'
'
 dir /r/kbasic, /r/ks.run
Introduction 'Instructions for Program
        PRINT CHR$(12)
        PRINT
        PRINT "THIS PROGRAM WILL GENERATE Y VALUES FOR ANY X"
        PRINT "A NUMBER OF POINTS FROM THE CURVE MUST BE ENTERED"
        PRINT "THEIR ORDER AND SPACING ARE NOT CRITICAL"
        PRINT
        PRINT "ENTER THE NUMBER OF DATA POINTS:"
        INPUT n%
        n%=n%-1
        DIM x(n%),y(n%,n%)
        GOSUB XYInput
        GOSUB  MakeDeltas
        PRINT
        INPUT "DO YOU WISH AN EQUATION?",answer$
        answer$=UPPER$(answer$)
        IF LEFT$(answer$,1)="Y" THEN GOSUB Equation
        PRINT
        PRINT "YOU CAN NOW FIND Y VALUES"
'
' COMPUTE Y'S FOR THE X
        LOOP
          INPUT LINE "ENTER AN X VALUE:",answer$
          answer$=UPPER$(answer$)
          EXITIF LEFT$(answer$,1)="Q" THEN PRINT "PROGRAM ABORTED"
         ENDEXIT
          x=VAL(answer$)
         GOSUB ComputeY
          PRINT "THE Y VALUE IS ....";f
         PRINT
        ENDLOOP
        END
'
'
XYInput 'Get X and Y values
        FOR i%=0 TO n%
          PRINT "ENTER X";i%;", Y";i%
          INPUT "", x(i%), y(i%,0)
        NEXT i%
        RETURN
'
ComputeY 'COMPUTE THE Y VALUE
        f=y(0,0)
        FOR j%=1 TO n%
          GOSUB MakePolys
          f=y(0,j%)*s+f
        NEXT j%
        RETURN
'
MakeDeltas ' Create Deltas for Y values
        FOR j%=1 TO n%
          FOR i%=0 TO n%-j%
             y(i%,j%)=(y(i%+1,j%-1)-y(i%,j%-1))/(x(i%+j%)-x(i%))
          NEXT i%
        NEXT j%
        RETURN
'
MakePolys 'CREATE POLYNOMIALS
        s=1
        FOR i%=0 TO j%-1
          s=(x-x(i%))*s
        NEXT i%
        RETURN
'
Equation ' Generate Polynonial
        PRINT "F(X)=";
        PRINT y(0,0);
        FOR i%=1 TO n%
          if y(0,i%)<>0 THEN
            PRINT "+";
             PRINT "(";y(0,i%);")";
            FOR j%=0 TO i%-1
              PRINT "(X-";STR$(x(j%));")";
            NEXT j%
          ENDIF
        NEXT i%
        PRINT
        PRINT
        RETURN

**  End of Listing
```



The Macintosh™ Section

Reserved as a

A place for your thoughts

Mac-Watch

READYSETGO

A DeskTop Publishing Page Make-up Program Version: 2.1

One of the more rapid growing aspects of our profession is *DeskTop Publishing*. Estimates are that the DeskTop Publishing segment of total USA microcomputer sales will exceed 60% of that total, for the coming several years. If so, then DeskTop Publishing will have a growth curve more pronounced than any other segment in our short history.

For the past several months, we here at 68 MICRO JOURNAL, have converted to setting our type with a Macintosh 68000 micro using several different publishing software programs. This review is one of the more popular programs - **ReadySetGo**.

ReadySetGo falls somewhere in the middle price range of programs of this type. Yet, it has nearly all the features of the more expensive programs. For most applications the cost difference belies the features of ReadySetGo as opposed to the more expensive programs. As a result most of you who plan on purchasing a DeskTop Publishing make-up program (for the Macintosh) will find this program is more than sufficient for most all your needs.

In doing a review of this type it is somewhat difficult to not attempt comparing peaches to breadfruit. Especially since we use both the more expensive programs (PageMaker, Stylo Formatter) and ReadySetGo. However, this review will be confined to ReadySetGo. It is not fair to attempt to compare a $195.00 program to a $495.00 program. *Fa t is, pri e wise, ReadySetGo is well worth the price.*

Another nice thing we like about this program is that it is not copy protected. We use an Apple hard disk on our system, and this program runs there fine, without having to verify our right to use the program by inserting an *'official'* disk each time we boot up and enter our publishing program. Where we have several persons using the system, and being booted several times a day, it becomes a real pain having to dig out the *'factory original'* and prove our honesty each time. Actually there are ways, using an un-copy protect program, several which are available, to get around this silly 'protection' scheme. But I think it is to their credit that the folks who published this particular program left us the right to make those much desired and necessary 'back-up' copies and also install the program on our hard disk without a lot of *'prove your honesty'* hassle! I might add that Stylo for the professional typesetter (we use it also) is not copy protected. *Most professional users have more important things to do than make up freebie copies for their friends.*

Another side observation. We use all the better publishing programs, for some application or another. Each has its strong points. But the folks at *Manhattan Graphics Corp*, the publishers of ReadySetGo have given us excellent support and seem to have a genuine interest in their customers. We started out with version 1.0 and the current 2.1 indicates a strong trend towards continual improvement.

One of the nice things about the Apple Macintosh is the *'What You See Is What You Get (WYSIWYG)'* way of doing things. All the page make-up programs do it to some degree or another. This program has several ways of displaying the screen and work area. One is a block display of each item block.

Blocks are created from the **CREATE** menu. There are six options in this menu. **TEXT - FRAME - SOLID - PICTURE** and **TEXT LINK.** Everything on the page must be placed within a block of its type.

The TEXT block as all the others is drawn to the screen as a one inch square. It then may be repositioned or resized as desired. While working in a TEXT block the menus FONT, STYLE and FORMAT are active, as in most all other Macintosh applications. The normal editing features are available, quite like in MacWrite.

The **TEXT LINK** function allows the linking of disconnected blocks to accept 'flowed' text. Text flowing is the process of calling a block of text created in some other editing mode (MacWrite for example) and then copying into a ReadySetGo application. Using the ability to create different size blocks and then placing them around graphic material,text can not only be flowed into linked blocks, but also around other blocks containing any other type material, by proper sizing and placing of linked text blocks around the material to be written around.

SOLID blocks have several purposes. One being highlighting, others include white on black block, heavy border blocks, uneven border block, etc. **FRAME** allows blocks with borders that may be toned in several degrees of dot toning. **PICTURE** blocks are used to contain pictures, graphs, etc. Again, everything done in ReadySetGo is placed in blocks.

Lines are developed from blocks that are resized to a thin area. Frame blocks can also make shaded lines. Lines can be defined in the **MODIFY** section by placement and size. Note discussion below as to sizes and precision.

Blocks are also addressed from another menu - **BLOCK**. The BLOCK menu has ten functions. **MODIFY** allows blocks to be sized to 1/1000th of an inch. FRAME blocks may have their border size set in the print trade form, points (72 points to the inch). Other blocks have slightly different options, depending on the type of block.

Generally block options are - in inches to a precision of 1/1000th inch - *Start Across, Start Down, Width, Depth,* etc. For example, the **TEXT** block has also the options of - *Para Indent, Left Indent, Tabs - On, Left and Decimal.*

The BLOCK menu also allows *Clearing, Duplicating, Deleting, Moving and Shift All* functions operate on all blocks. Also the BLOCK menu allows going directly to *Next Block, Previous Block, First Block and Last Block.*

The **PAGE** menu has three functions. Insert a new page, this can be blank or a duplicate of current page. Insertion can be before or after the current page. Page numbers are reodered automatically. *Delete* allows the deletion of an existing page. Also there are functions for scrolling through the pages of the document you are working on.

The **SPECIAL** menu has five functions. *Show Page* shows the entire page in block form. This reduced display may be left on the screen and allows for precise positioning of various blocks on the page. It is updated as blocks are moved, expanded, contracted, etc. In this mode no text or graphics are displayed, just the blocks and their accurate position on the page, which is grid overlaid (may be turned off) for precise placement. *Hide Grid/Show Grid* is a toggle function. *Send Behind* is the function that allows operations on blocks that overlay or overlap. Blocks may be sent to the back as required to display what is underneath. *Display Page* is also a reduced page display. However, it cannot remain on the screen while other operations are being performed. It does display all the text, picture, graphs, etc. *Display Facing Pages*, is a function that displays mini-pages of the document in facing page order. This function also must be removed from the screen when other operations are being done.

ReadySetGo supports both the ImageWriter and newer professional LaserWriter.

An additional DA (desk accessory) called ArtGrabber+ is included, at no additional cost. **ArtGrabber+** is licensed from *MacroMind*. ArtGrabber+ allows the users to import MacPaint files directly into ReadySetGo. Also the entire page or any portion can be selected. ArtGrabber+ puts the selected material into the clipboard and it is used as any other clipboard item. We like ArtGrabber+. It makes a nice supplement to an already good program.

The documentation for ReadySetGo is over 100 pages and is, as you would expect, professionally done. Even the least professional of us users should have no fear of this program. While it is a very powerful DeskTop Publishing program, it is very user friendly. We like that. We, as was said before, also like the quality of the support offered. All in all, for the price, ReadySetGo is a real bargain at $195.00.

EOF

A Staff Review

Beginning To C

By J. A. (Jim) Woodward

Background

I am a math instructor who owns a color computer with 2 ss dd 40 track disk drives, a J & R 256k Banker, and WordpakII. Stylograph is the word processor that I use. Knowing Basic09 to some extent, I decided to learn another language. It is my opinion that Pascal is very similar to Basic09, so I chose C as the next language to learn. I must admit, however, that if it were not for Dale Puckett's books *(recommended Reading - DMW),* I would still be using my computer to write letters and play KING.

I first obtained the C compiler from Radio Shack and then obtained a book titled: *A BOOK ON C* by Al Kelly and Ira Pohl, and started messing around. It took me a week to find out how to control the cursor in a C program. I then thought up a simple board game to demonstrate cursor control. The following game has a playing board consisting of 20 rows and 78 columns, thus there are 20*78=1560 locations on the board. The coordinates of a location are: a row followed by a column, e.g. 20,78 would represent the lower right corner of the board, and 1,1 would represent the upper left corner. I think this game will run on any 80 column 6809 computer, if one fixes the functions crs() and flip() to go with their terminal. I am writing this article hoping to find this to be the case. For if this is the case, then many programs could be written in C, the special functions denoted with comments, and people could replace them, if required to, with functions designed for their terminal.

In my program the crs(x,y) puts the cursor on row x, column y. I am using WordpakII which has a 24 row (0 through 23) screen. The program runs also on my son's 25 row screen using Wordpak, if he remembers to enter:< tmode type=80 > before trying to play the game.

I think the game is totally debugged, as my sons can't foul it up, no matter what they enter. Showing this game to people that dropped by (Ò They had no other choice but to try it!), I learned that it is quite helpful in teaching people about using coordinates to locate a position. Sometimes things that are obvious to one person are not so obvious to another.

The GAME

The object of the game is to locate a position predetermined by the computer. You choose your firing coordinates and an X appears on the board at those coordinates. A @ marks each of the previously fired upon positions. If they are the winning coordinates, you are so notified. If they are not the winning coordinates, a number (shot value) representing the distance your coordinates are from the winning coordinates appears in the header. The distance is the sum of the squares of the row error and the column error, thus as you get closer your shot value will decrease. Your score is increased each shot by the shot number times the shot value. If more than one person is playing, then the lowest score is the winner. In some ways, this is a good game as the winner is often the person with the best first guess,but not always. This teaches one that luck plays an important part of almost everything, but in the long run, people who are prepared get rewarded to a larger extent than those who rely solely on luck.

In conclusion, I wrote this hoping it might help some beginners, like me, to understand some C programming concepts. I need to see many simple programs in order to understand the concepts involved, for example, I am just beginning to understand about pointers. I was quite proud of my seed() function as I know nothing about an assembly language. I should imagine that a real C programmer will cringe when and if they read my GAME, as I'm sure there are better ways to do most everything I've done, but just remember: this was my first C program.

+++

Editor's Note: Thanks for the above. We need more beginner type input. For all languages! We are gaining new readers every month. Many are beginners. They (and probably all of us) need this type of information. How 'bout it?

DMW

```
list 3.game.c

/*  This program was written on a Color Computer with Wordpak II   */
/*  using the RS display mode. It will run with Wordpak, but be    */
/*  sure you are in the RS display mode  (enter  tmode type=80)    */
/*  The Microware C compiler was used to compile this program      */

#include <stdio.h>
#include <os9.h>

main()
{
int    n, wplyr,plyrs,shot, num, erx, ery, wc, wr, x, y, crs(), flip(),
int    i, ch, seed(), rand(),seedx, seedy, hx, hy;
long   score, wscore;
char   str[10],answer, board();

pflinit();
new   wscore=2000000000;
plyrs=0;
seedx=seed(97),
wplyr=0,
putchar('\xc').
crs(10,20),
puts("Enter the number of players, please  ").
crs(11,31),
puts(" 1, 2, or 3");
while (plyrs!=1 && plyrs!=2 && plyrs!=3) {
      crs(11,50),
      plyrs=getchar()-'0';
      }
for (n=1,n<=plyrs,++n)  {
   seedy=seed(97);
   board(n),
   wr=(rand(seedx)%20)+3;
   wc=(rand(seedy)%78)+1;
   shot=-1,answer='q';
   hx=hy=score=num=0;
  while (shot!=0) {              /* start of game*n   */
     if (shot==-1) {
     crs(0,20),
     puts("New Game, enter first shot, please ");
     }
  redo: crs(0,0),
   i=0,
   y=99,
   str[0]=NULL,

while ((ch=getchar())!=EOL && i<9) {
 if ((ch>='0' && ch<='9') || ch==' ' || ch==',')  {
  if (ch==',')
   ch=' ',
  str[i++]=ch,
  }
}
str[i]=NULL,
sscanf(str,"%d%d",&x,&y),
if (num>0)  {
crs(hx,hy).
putchar('●'),
}
if (x>20 || x<1 || y>78 || y<1)  {
 if (shot!=-1) {
  crs(0,0),
  puts("        Reenter your coordinates, please  "),
  }
 if (shot==-1 && answer=='z') {
  crs(0,0),puts("        "),
  crs(0,46),puts(" again, please "),
  }
  answer='z',
  goto redo,
}
crs(0,60),
putchar('\3'),
printf("(%2d,%2d)",x,y),
crs(0,0),
x= x+2,
crs(x,y),
putchar('X'),
hx=x,hy=y,
++num,
erx=(wr-x)*(wr-x),ery=(wc-y)*(wc-y),
seedy=seed(97),
        shot=erx+ery,
        score+=((long)num*(long)shot),
        crs(1,10),
        putchar('\3'),
        printf("Shot(%d)= %d",num,shot),
        crs(1,50),
        printf("Score = %ld",score),
        crs(0,0),
        puts("?      : Enter shot coordinates "),
        }  /*  endwhile, end of game*n  */
      for (x=1,x<22,++x) {
      flip(1),
      crs(12,30),
      puts("WINNER--WINNER"),
      flip(0),
      crs(12,30);
      puts("WINNER--WINNER"),
      }
    seedx=seed(97);
   if (wscore==score)
      wplyr=wplyr*wplyr+n*n,
   if (wscore>score) {
      wscore=score,
      wplyr=n,
      }
   }  /*  end of for(n) loop  */
 putchar('\xc'),
 crs(9,20),
 if (wplyr>13)
 printf("Players 1, 2, and 3 tied with scores of %ld ",wscore),
 else {
 if (wplyr>10)
 printf("Players 2 and 3 tied with a score of %ld ",wscore),
 else {
 if (wplyr>5)
 printf("Players 1 and 3 tied with a score of %ld", wscore),
 else  {
 if (wplyr>3)
 printf("Players 1 and 2 tied with a score of %ld ", wscore),
 else {
 if (plyrs>1)
 printf("Player %d won with the score of %ld",wplyr,wscore),
   }
   }
   }
   }
 crs(11,20),
 puts("Would you like another game?  < y , n > "),
 while (1) {
    crs(11,62),
    ch=getchar(),
    if (ch=='y')
       goto new,
    if  (ch=='n')
       exit(0),
  }
 } /*  end of game    */
 int  crs(r,c)        /*  This function is for the CC with Wordpak  */
   int   r, c,    /*  In the RS display model               */

   {
   putchar('\x1');
   r%=24, c%=80,
   r+=32,c+=32,
   putchar('\x2');putchar(c),putchar(r),
   }
int flip(x)      /*  This function is for the CC with Wordpak   */
  int  x;      /*  in the RS display model                */
```

```
{
  if (x==0)
     x=32,
  else
     x=33,
     putchar('\x1b');putchar('\x53');putchar(x),
}
char board(n)
  int  n,

  {
  int  x, p,

 putchar('\xc'),
 flip(1),
 crs(23,79),
 putchar('*'),
 for (x=3;x<23,++x) (
    p=(x-2)%10,
    crs(x,0),
    printf("%1d",p),
    crs(x,79),
    printf("%1d",p),
    }
 for (x=1,x<79,++x){
    crs(23,x),
    p=(x%10),
    printf("%1d",p),
    crs(2,x),
    printf("%1d",p),
    }
 crs(2,0),
 putchar('*'),
 crs(2,79),
 putchar('*'),
 crs(23,0),
 putchar('*'),

   flip(0),
   crs(1,30),
   printf("Player %d ",n),
   }
 rand(x)
   int   x,

   {
   static  int   n;

   n=(25173*n*x+13849)%65536,
   n=(n<0) ? -n: n,
   return(n);
 }
 int  seed(x)

    int  x,
 {
   int k, i, n, m,
   struct  registers  reg,
         cnar  *p, time[9],

   x=(x<0)?-x:x,
   p=time,
   reg.rg_x=p,
   _os9(F_TIME,&reg),
   n=((time[4]+time[5])*time[5])%x;
   n++,
   return (n), .
 }

 +++
```

FORTH
Is Fast

MACH1 (TM), A MULTITASKING FORTH DEVELOPMENT SYSTEM FOR THE GESMPU-4A 68000 CPU.

By:
Aleksey Novicov and Henry Sharpe
Palo Alto Shipping Co.

This article discusses FORTH and the benefits and use of a flexible development environment for Motorola 68000 based computers that includes:

Mach1 Forth (TM), an EPROM based Forth Kernel that is resident on the computer being developed (often called the target). TargetTerm (TM), a terminal emulator program that runs on the Macintosh that permits transfer of files to or from the target via a serial port.

A Target Computer, in this case, the *GESPAC GESMPU-4A*, and whatever G-64 peripheral boards are needed for a particular application. *The GESMPU-4A was chosen because its a 68000 based board that has room for 256K of EPROM, has a serial port, is physically very small, and has an unrivaled price performance ratio.*

INTRODUCTION

This system was conceived and implemented by a multidisciplinary team of engineers to permit efficient, quick turn-around time for the development of microprocessor based products, particularly those used to control real-time applications. Several ideas are central to our strategy and are reflected in the software itself.

1. We feel that a development environment should be interactive. With Mach1 Forth, ideas can be developed very quickly by simply typing them in, and running them immediately. Assembling and linking are not required and consequently working results are achieved without an interruption to the programmer's thought process. This fosters an iterative design approach; it's so easy to try out a new idea that programmers are likely to strive for something clearer and more elegant than the first thing that happens to work.

2. Generally, all code should be run on the target itself, not on a host computer emulating a target. This is particularly true for target computers that are controlling real-time systems such as mechanisms or communications hardware. In the case of a computer controlled mechanism, for example, the programmer becomes aware of any unforeseen dynamic behavior the system may exhibit from the start of system integration thereby avoiding any surprises late in the product's evolution.

3. The host computer supplies resources that most targets have no need for: disk drives, CRTs and full keyboards. Targets may have use for these devices, but particularly in the development phase it's prudent to rely on the tested resources of the host. Therefore, source code is stored and edited on the host while communications and file transfer between host and target are handled by running a terminal emulator on the host.

4. Multi-Tasking simplifies Real-Time Applications. Frequently the CPU must control several unrelated tasks. Multi-tasking allows the programmer to think about such unrelated tasks independently. Mach1 provides two kinds of tasks: fully asynchronous Terminal Tasks for user I/O, and Background Tasks that are scheduled on a round robin basis. This approach is designed to minimize the programming complexity and overhead required to manage multi-tasking, and also to offer the degree of system responsiveness necessary to control real-time events.

WHAT IS FORTH ?

The foundation of Forth is the "Forth dictionary". The Forth dictionary is exactly that: a collection of words that define a language. Each word in the Forth dictionary, when called upon, performs a specific action. This action may be as simple as moving a number from one part of computer memory to another, or as complex as writing data to a floppy disk. This collection of predefined words is called the "Forth kernel."

Writing a program in Forth simply consists of defining a new word in terms of words that already exist in the dictionary. Once a new word has been defined, it is added to the dictionary and can subsequently be used to define other words. Let's look at an example:

```
: STAR       42 EMIT ;  <cr>ok <0>
: 3STARS  STAR  STAR  STAR  ;  <CR>ok <0>

STAR <cr> * ok <0>
3STARS <cr> *** ok <0>
```

We start the definition of a new word with a colon. The colon indicates to Forth that we are about to define a new word. The name which immediately follows a colon (STAR in the example above) will be the name of the new word. A semicolon marks the end of the new word's definition. Any words between the name and the semicolon determine what the new word will do.

STAR's actions will be 42 EMIT. EMIT is a word predefined in Forth's kernel that prints out the character corresponding to the ASCII value passed to it. Since 42 is the ASCII value for an asterisk, when star is executed an asterisk will be printed to the screen.

After STAR is defined it becomes part of the Forth dictionary and can be used in the definition of other words such as 3STAR. 3STAR's action will be to print 3 consecutive aster sks to the screen. The ability of a computer language to extend itself is called extensibility. This is one of the many desirable features that is an integral part of Forth.

FORTH IS AN INTERACTIVE LANGUAGE

If in the above example, you had only typed 42 EMIT , an asterisk would have been printed to the screen. You would have interactively executed the word EMIT. But what happened when you defined STAR in the example above? Why was no asterisk printed to the screen when you typed EMIT then? In Forth there are two modes of operation:

1) Compiling mode, which is invoked by a colon to indicate you are defining the actions of a new word.

2) Interpreting, or execution mode. In Forth, whenever you hit the carriage return and get the "ok" prompt, you know you're in the execution mode. In this mode you can interactively execute any word, or sequence of words that exist in the dictionary.

After you hit the carriage return Mach1 Forth interprets the words you typed in and immediately executes the compiled code associated with each word definition.

The power of an interactive language is that you can quickly try out any sequence of words that you've defined, and if necessary, modify the words and try them out again in seconds. Many interactive languages are S L O W because they are required to always interpret code. Forth however, compiles most code (for example the code inside colon words) and consequently can run considetably faster. Due to an internal improvement, Mach1 Forth runs several times faster than most other Forths, and at normalized speeds, comparable to TurboPascal and compiled C.

FORTH IS FAST

The reason this version of Forth can run so quickly is that it is "subroutine threaded." What this means is that each word in the dictionary is actually compiled (i.e. stored in memory) as an assembly language subroutine. For example, if you were defining a new word from predefined kernel words it would consist of a series of JSR (assembly language Jump to Sub Routine) calls to each kernel word. In turn each word in the kernel consists of the machine code instructions appropriate to its function. Usually the programmer does not have to worry about these issues; he writes code in a high level language with the confidence that it will be able to run quickly.

However there are instances where it's appropriate to write code directly in assembly language. Fortunately Mach1 Forth includes a built-in standard 68000. Imagine you are interfacing a new motor controller chip to the system. A sample assembly language initialization is included with the chip's data sheet. To define a new assembly language word that will run the code from the data sheet type:

```
CODE InitMotor
      < Enter the assembly language code >
      < included with the data sheet here.  >
END-CODE
```

"Code" indicates to Forth that we are about to define an assembly language word. "InitMotor" is the new word's name. Assembly language mnemonics follow the name, and finally "End-Code" signals that the assembly language definition is complete. Typing "InitMotor" will immediately execute the exact code included with the data sheet and it will run at full processor speed. It's this easy to include assembly language routines in Mach1 Forth.

FORTH HAS A STACK

Having a collection of words that perform certain actions is fine, but how do these words communicate or pass data? The answer is through what's called the "parameter stack", a data structure also known as a LIFO (Last-In-First-Out) queue. Some words put data on the stack, others remove it, some do both, and still others don't require the stack at all.

An example of a word that takes data from the stack and also leaves a result is the math operator "+". This Forth word called "plus" takes 2 numbers from the stack, adds them, and leaves the result on the stack. The Forth word "." , which is pronounced "dot" prints the number on the stack to the screen:

```
34  66  +  . <cr>  100  ok
50  70  -  3 + .  <cr>  -17  ok
```

This method of arithmetic calculation is called Reverse Polish Notation, (RPN) and is used by Hewlett Packard calculators. The Forth kernel includes words to directly manipulate the stack so numbers can be arranged in the proper sequence for a particular operation: for example, DUP duplicates the top number on the stack, and SWAP swaps the position of the top two numbers. However Mach1 Forth includes two powerful constructs that eliminate the need for the programmer to directly juggle numbers on the stack. These constructs are called Named Input Parameters and Local Variables.

The "T" starts the list of local variables. Local variables are variables which will only be recognized inside the current colon definition, but will be unknown elsewhere. In effect they provide a temporary, "scratchpad" variable.

After the declaration is stated, the programmer simply specifies the name of an input parameter when that value is required for a calculation within the current definition. Local variables are used identically, but are not initialized to a particular value upon entry to the definition. The " -> " symbol takes a value off the stack and stores it into the following local variable or named input parameter. Amazingly, in addition to both improving the readability of the code and making it easier to write, no speed penalty is suffered through the use of these two constructs. Another thing to notice is that in Forth, the names' length is not limited and therefore words can be chosen to fully describe the word's action, further easing the program's readability.

MACH1 CAN MULTI-TASK

The multi-tasker in Mach1 is simple to use but offers a high level of performance. Although terminal I/O is fully asynchronous, most tasks switch context on a round-robin basis. What this means is that all the tasks are linked together in a circular list and the CPU simply moves around the list. Context is switched from one task to the next by execution of the Forth word PAUSE. Because Forth is stack based, context switching happens extremely quickly: on the order of 6 microseconds as opposed to 130 microseconds for other 8 MHz 68000 based real-time executives. The programmer can easily pass control away from less important tasks by including PAUSE at strategic intervals in the code. Conversely, important tasks PAUSE only once.

Without detailing a full example, new tasks can be easily implemented. Approximately 3 short lines of code is all that is required to set up and control a new task. BUILD builds a new task, ACTIVATE turns it on. By using WAKE and SLEEP tasks can be woken up or put to sleep, either by themselves or by other tasks. GET and RELEASE control semaphores that channel task access to limited system resources such as disks or printers. A final feature is that all the words in the Forth dictionary are accessible to each task. This enables two separate tasks to share routines which eliminates the need to have redundant code. Shared routines must be reentrant and Mach1 are naturally reentrant when using local variables. Using global variables, message queues and "mailboxes" for inter-task communication can be readily implemented. Finally, events in tasks, or even tasks themselves can be interrupt driven.

FORTH IS A STRUCTURED LANGUAGE

In any kind of program different actions are taken depending on external input, whether this input comes from a user at a keyboard, or from a photodiode sensor attached to a computer's input port. In order to determine what actions to take decisions have to be made. Standard control structures exist in Forth to regulate the flow of a program, thereby forming the basis for structured programming.

```
These structures include:
1)  If....Then
2)  If....Else....Then
3)  Begin....Until
4)  Begin....While....Repeat
5)  Do....Loop
```

☎ (615)842-4600 Telex 5106006630

South East Media

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

SPECIAL

K-BASIC

K-BASIC under OS-9 and FLEX will compile TSC BASIC, XBASIC and XPC Source Code Files.

$AVE
$100

K-BASIC now makes the multitude of TSC XBASIC Software available for use under OS-9. Transfer your favorite BASIC Programs to OS-9, compile them, Assemble them, and BINGO -- useable, multi-precision, familiar Software is running under favorite Operating System!

!!! SPECIAL ~~$199.00~~ $99.00 !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE

With every development system you receive:

□ A manual that makes sense
□ A periodic newsletter
□ Screen form language
□ Report generator
□ Menu system
□ Query facility
□ Set of utility programs
□ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts ■■■■■ **Features** ■■■■■■■■

DATA DICTIONARY

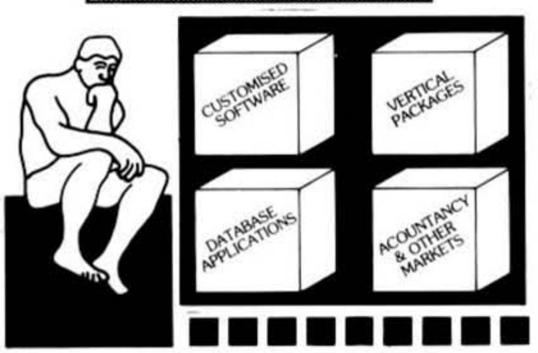
Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

□ alphanumeric
□ integer
□ floating point
□ money
□ date

DATA FILE STRUCTURE

□ Packed, fixed-length records
□ Money stored in lower currency unit
□ Dates stored as integer day numbers

Sculptor for 68020 OS-9 & UniFLEX $995



INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

□ automatic by field type
□ validation list in data dictionary
□ programmer coded logic

ARITHMETIC OPERATORS

–	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
–	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES

□ Full date arithmetic
□ Echo suppression for passwords
□ Terminal and printer independence
□ Parameter passing to sub-programs
□ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS

□ Define record layout
□ Create new indexed file
□ Generate standard screen-form program
□ Generate standard report program
□ Compile screen-form program
□ Compile report program
□ Screen-form program interpreter
□ Report program interpreter
□ Menu interpreter

□ Query by date
□ Reformat file
□ Check file integrity
□ Rebuild index
□ Alter language and date format
□ Setup terminal characteristics
□ Setup printer characteristics

SCREEN-FORM LANGUAGE

□ Programmer defined options and logic
□ Multiple files open in one program
□ Default or programmer processing of exception conditions
□ Powerful verbs for input, display and file access
□ Simultaneous display of multiple records
□ Facility to call sub-programs and operating system commands
□ Conditional statements
□ Subroutines
□ Independence of terminal type

* Full Developement Package
** Run Time Only
*** C Key File Library

OS-9/UniFLEX IBM PC Zenix MS DOS Network	--	$995 / $199 / $498 • • •••	
68000 UniFLEX Altos Zenix UNIX	--	$1595 / $319 / $798 • •• •••	
MS DOS PC DOS	--	$595 / $119 / $595 • •• •••	

MUSTANG-020™ Users - ask for special discount.

Sculptor is a Trademark of Microprocessor Developments Ltd.

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends

F – FLEX, CCF – Color Computer FLEX
O – OS-9, CCO = Color Computer OS-9
U – UniFLEX
CCD – Color Computer Disk
CCT – Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

South East Media
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Fore n

VISA

ASSEMBLERS

ASTRUK09 from S.E. Media – A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.

F, CCF - $99.95

Macro Assembler for TSC -- The FLEX STANDARD Assembler.

Special -- CCF $35.00; F $50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. GeneÔrate OS-9 Memory modules under FLEX.

FLEX, CCF, OS-9 $99.00

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.

F, CCF $150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.I.. Programming for small to medium-sized Programs.

F, CCF - $75.00

XMACE – MACE w/Cross Assembler for 6800/1/2/3/8 F, CCF - $98.00

CROSS-ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants - - Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.

FLEX, CCF, OS-9, UniFLEX each - $50.00
any 3 objects or source (not 68000) - $100.00
Set of ALL object $200.00 - source $300.00
UniFLEX 68000 - $50.00 - source $1,000.00

XASM Cross Assemblers for FLEX from S.E. MEDIA – This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.

Complete set, FLEX only - $150.00

CRASMB from LLOYD I/O – Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties): 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX(binary). Written in Assembler ... e.g. Very Fast.

Availability:	MOTOROLA	INTEL	OTHER	COMPLETE SET
	CPUs	CPUs	CPUs	CPUs
FLEX9	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.

FLEX, CCF, OS-9/6809 $249.00

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

O & CCO obj. only – $39.95; w/ Source - $79.95

Lucidata PASCAL UTILITIES (Requires LUCIDATA Pascal ver 3)

XREF – produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE – Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

F, CCF -- EACH 5" - $40.00, 8" - $50.00

☎ (615)842-4600 Telex 5106006630

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

DUB from S.E. Media – A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.

U - $219.95

LOW COST PROGRAM KITS from S.E. Media – The following programs are available for FLEX on either 5 or 8 inch disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX UTILITIES KIT $99.95**
   CATS.CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESO.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization where required
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer program
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.

F and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F – FLEX, CCF – Color Computer FLEX
O – OS-9, CCO = Color Computer OS-9
U – UniFLEX
CCD – Color Computer Disk
CCT – Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

**** Shipping ****

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

Telex 5106006630
☎ (615)842-4600
South East Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No "blind" debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!

Levels I & II only - OS-9 Regular $149.95
SPECIAL INTRODUCTION OFFER *$69.95*

DISK UTILITIES

OS-9 VDisk from S.E. Media – For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.

Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media – Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX Directory, Delete FLEX Files, Copy both directions, etc. FLEX users use the special disk just like any other FLEX disk

O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.

OS-9 $85.00

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX disk (8 - 5 - hard disk) can have sub directories.** By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

"Introduction Special" $69.95

COPYMULT from S.E. Media – Copy LARGE Disks to several smaller disks. FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, 8" or 5") $99.50

COPYCAT from Lucidata – *Pascal NOT required.* Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

F and CCF 5" - $50.00 F 8" - $65.00

FLEX DISK UTILITIES from Computer Systems Consultants – Eight (8) different Assembly Language (w/ Source Code) FLEX Utilities for every FLEX Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). – PLUS - - Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or A.L. Source Code).
F and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. – Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, CCF, OS-9, UniFLEX; with complete
Source $100.00 without Source $50.00
UniFLEX 68000 with complete Source $100.00

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F – FLEX, CCF – Color Computer FLEX
O – OS-9, CCO = Color Computer OS-9
U – UniFLEX
CCD – Color Computer Disk
CCT – Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

South East Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

" Shipping "

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) $99.95
X-TALK Software (2 disks only) $69.95
X-TALK with CMODEM Source $149.95

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.

U - $299.99

EDITORS & WORD PROCESSING

JUST from S.E. **Media** -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

* Now supplied as a two disk set:
Disk #1: JUST2.CMD object file, JUST2.TXT PL9 source FLEX - CC
Disk #2: JUSTSC object and source in C: FLEX - OS9 - CC

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only - F & CCF - $49.95
Disk Set (2) - F & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. **Media** - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX $129.50
"SPECIAL INTRODUCTION OFFER" $79.95
SPECIAL PAT/JUST COMBO (w/source)
FLEX $99.95
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

Regular $129.95
SPECIAL INTRODUCTION OFFER FLEX $69.95

☎ (615)842-4600 Telex 5106006630

South East Media

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, STAR-DOS Regular $69.95
Limited Special Offer: $39.95

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" column (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

SPELLB "Computer Dictionary" from S.E. **Media** -- OVER **150,000** words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/STAR-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,
F or O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,
F or O - $99.95, U - $149.95

STYLO.MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - $59.95,
F or O - $79.95, U - $129.95

STYLO.PAK --- Graph + Spell + Merge Package Deal!!!

F or O - $329.95, U - $549.95
O, 68000 $595.00

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends--

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

South East Media

5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

☎ Telex 5106006630
(615)842-4600

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems – By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

F, CCF - $198.00

PASC from S.E. Media - A Flex9 Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.

FLEX $95.00

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers*. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

F and CCF - $195.00

FORTH from Stearns Electronics – A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - $58.95

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

F and CCF - $295.00

C Compiler from Introl – Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.

FLEX, CCF, OS-9 (Level II ONLY), U - $575.00

PASCAL Compiler from Lucidata – ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $99.95 F 8" - $99.95

PASCAL Compiler from OmegaSoft (now Certified Software) - - For the *PROFESSIONAL*; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.

F and CCF - $425.00 - One Year Maint. $100.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA – A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

FLEX, CCF, OS-9 Compiler/Assembler $199.00

CRUNCH COBOL from S.E. MEDIA – Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*

FLEX, CCF; Normally $199.00
Special Introductory Price $99.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media – Requires FLEX and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*

F and CCF - $79.95

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00

DYNAMITE+ – Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00 - CO, Obj. Only $59.95
F, " " $100.00 - O, object only $150.00
U, " " $300.00

DATA-BASE ACCOUNTING

XDMS from Westchester Applied Business Systems - Powerful DBMS; M.L. program will work on a single sided 5" disk, yet is F-A-S-T. XDMS Level I provides an "entry level" System for defining a Data Base, entering and changing the Data, and producing Reports. XDMS Level II adds the POWERFUL "GENERATE" facility with an English Language Command Structure for manipulating the Data to create new file Structures, Sort, Select, Calculate, etc. XDMS Level III adds special "Utilities" which provide additional ease in setting up a Data Base, such as copying old data into new Data Structures, changing System Parameters, etc.

XDMS System Manual - $24.95
XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95

XDMS IV from Westchester Applied Business Systems - *XDMS IV* is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, *but also to process entire files producing customized reports, screen displays and file output.* Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotalling, and presentation of up to three related files as a "database" on user defined output reports.

*XDMS IV - F, CCF STAR-DOS, SK*DOS $350.00*
Upgrades to XDMS IV - $250.00

☎ (615)842-4600

Telex 5106006630

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants – TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.

F, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants - - The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media – An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

F - $59.95, U - $89.95

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends--

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

**** Shipping ****

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

MasterCard

THE FORTH DEVELOPMENT CYCLE

There are basically three steps in the development of a Forth based product:

1. Configuring the kernel to the target computer, thereby allowing the application to be written in high level Forth.

2. Developing an application written in Forth that runs on the target in RAM.

3. Converting the RAM based application to EPROM. This system assumes a host computer, probably (but not necessarily) an Apple Macintosh (TM), and a Motorola 68000 based target computer that includes a serial port.

CONFIGURING THE KERNEL

If the Gespac GESMPU-4A board is the desired target a working preconfigured kernel is available, and the user needs to do nothing more than attach a terminal to the board's serial port and as soon as power is applied, the familiar "ok" Forth prompt will appear. However, the user has the capability to reconfigure the memory map because, for instance, RAM and EPROM will reside at different addresses than within the preconfigured kernel. To do this, the precompiled Forth kernel and an example User I/O file are needed. These are available through Palo Alto Shipping Company. The information in the User I/O file tells the kernel the memory location of RAM, ROM, and the target's serial port. It also includes a simple assembly language driver for the target computer's serial port that can send a character, and receive a character.

DEVELOPING AN APPLICATION

At this point new words can be added to the dictionary and the full resources of the kernel are available. Typically however there are other peripheral boards to get working: parallel ports, other serial ports, motor controllers, to mention a few. These peripherals usually each require some initialization and low level routines from which the higher level routines will be built. For instance to continue with the motor example from above, a few words to make the motor turn in a specified direction, for a given distance and speed would probably be needed.

In instances like this Forth shines because the programmer can interactively determine how the controller works. If something doesn't quite work as expected the first time, you're able to try something a little different very quickly due to Forth's interactive nature. This iteration continues until the programmer is comfortable with an understanding of that part of the system's behavior.

This is where the "development environment" comes into play. It consists of an Editor, TargetTerm and Switcher (TM) which are all parts of the development package for the Apple Macintosh. As a group these programs allow the user to write code, save it on the host's disks, download it to the target where it's compiled, and then interactively test out the new words on the target.

The new definitions can be tried out simply by typing their names. These trials either indicate that the new words work properly and therefore will legitimately be able to be included in still higher level words, or that there's a problem. If a problem is revealed, the programmer can easily execute the underlying words until the exact culprit is found. A nice feature of most of the Gespac RAM boards are write protect switches, so that in the event of a system crash the reset button is pushed and you're instantly back to where you were before the crash. In this manner the programmer can easily isolate where code must be modified and the edit, download, test cycle is repeated.

This cycle is set up to occur fluidly to minimize the disruption of the programmer's concentration. The fluidity of this cycle is important because it is repeated many times until a complete application is developed, at which point you're ready to burn EPROMs to complete the product.

CONVERTING TO EPROM

Usually a target computer runs a dedicated application. Examples are the computers inside printers, plotters or a copier. The last word in a copier application might look like this:

```
: COPIER
         INITIALIZE
         BEGIN
         Get_User_Input
         Set_Exposure
         Set_Paper_Size
                             Make_Copies
                   AGAIN  ;
```

Notice that the word is an endless loop. It will look for user input, make copies, then look for user input again. The only problem is that this word should execute immediately and automatically after power up. This is easily accomplished using utilities in Mach1 which enable the user to have the last word defined (in this case COPIER) to auto-execute upon power-up. Another feature to note is that Forth automatically makes the distinction between:
1. Program instructions, which do not change and hence can exist in EPROM, and
2. variable data, which does change and consequently must exist in RAM.

In other words memory is already organized in such a way that EPROMs can be burned immediately with no special steps required.

Finally we are ready to take the last step. The application is uploaded and the resulting file is sent to an EPROM programmer using utilities in TargetTerm. The RAM memory which used to hold program code is replaced by the new Application EPROMs and the development cycle of a Forth application is complete.

Further questions on this FORTH package should be addressed to Rick Mialey at Palo Alto Shipping Co., PO Box 7430, Menlo Park, CA 94026, (408) 854-7994.

NOTES:
Macintosh is a registered trademark of Apple Computer Co. which also owns the copyrights to Switcher and The Macintosh Development System (MDS) which includes EDIT.

Mach1 is a registered trademark of Palo Alto Shipping Co.

Author's biographies:

Aleksey Novikov and Henry Sharpe are both mechanical engineers holding Masters Degrees from Stanford University. Their expertise lies in the computer control of electromechanical devices and the user-interface to products incorporating such systems. Both individuals have participated in teaching a graduate level course in Smart Product Design at Stanford and have developed several Forth based products. Aleksey was a contributor in the development of Mach1 Forth, originally written for the Macintosh. He and Henry, in conjunction with Rick Miley, Terry Noyes and Lori Chavez of Palo Alto Shipping Co. have recently taken that kernel and adapted it for use on target computers as described in this article. Both Aleksey and Henry currently work as independent consultants.

EOF

MIKEINTOSH

By: Mike Woll
Atomic City Electronics
901 18th #115
Los Alamos, NM 87544

Do you want to get a MacIntosh but can't afford it, or are you having trouble driving the kids/spouse away from the Mac so you can finish your game of ZORK? Maybe a Mac is the answer. But another $2k is too much. Well there's a way if you're a little handy with electronic stuff or have a friend who is.

The first thing is to obtain a Mac Logic Board, keyboard, and disc drive. There are a surprising number of these floating around. Many people upgraded to 512K by ordering a board from Apple and still have their 128K board. Others have upgraded to the new 800k drives or to a Mac Plus. The going price seems to be $70 for a 128K board, $225 for a 512K board, $75 for a disk drive. Keyboards might be found if someone got the new Mac Plus keyboard. Ask around your local Mac Users Group. If all else fails you might persuade your local dealer to sell one he removed from a Mac upgrade.

You will need a board, a disc drive, a power supply, a keyboard, a mouse, a box to put it in, and a suitable monitor.

Disc Drive: If you can't find a used upgraded drive, a standard external drive will work just fine except for a few programs which expect the "master" disc on the internal drive. The only one I have found so far is Lode Runner. This may be a problem if you're building it for the kids to play games on. However if you have a second drive that you can spare it is the cheapest solution. If you have to buy the drive, several people offer external drives at prices below $300. You also could buy the 800k upgrade kit. Discount price is about $350. An external drive can be converted to look like an internal drive by connecting it to the Mac board with a 20 pin ribbon cable from the internal drive connector instead of the external drive connector.

Power Supply: You will need a power supply that supplys at least +5v @ 2.5A, +12v @ 2A, and -12v @ 100ma. These are avaliable from many sources for prices of $30-$50. Data-Comp has a suitable one.

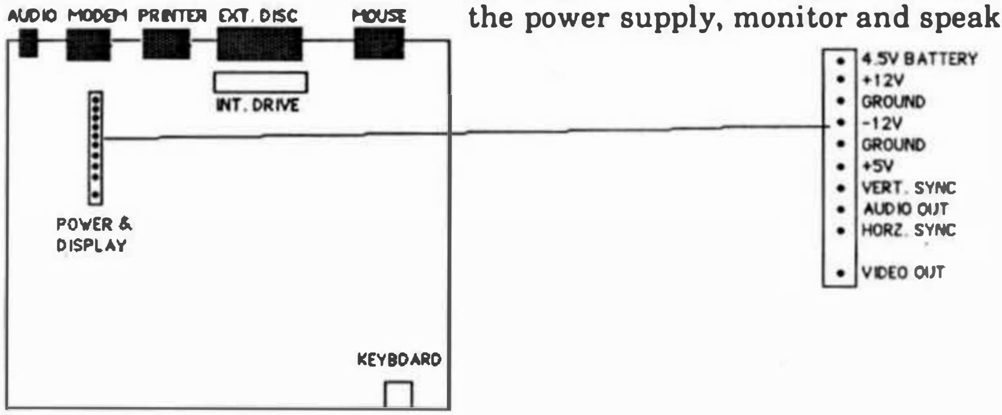
Keyboard: First try to find a upgrade leftover. I would think it should sell for <$50. After much searching I settled for a replacment keyboard from Apple at a list of $125. If you know your dealer you might get it for less. Dealers cost is around $80. If you have to go this route you might as well get the Mac Plus keyboard.

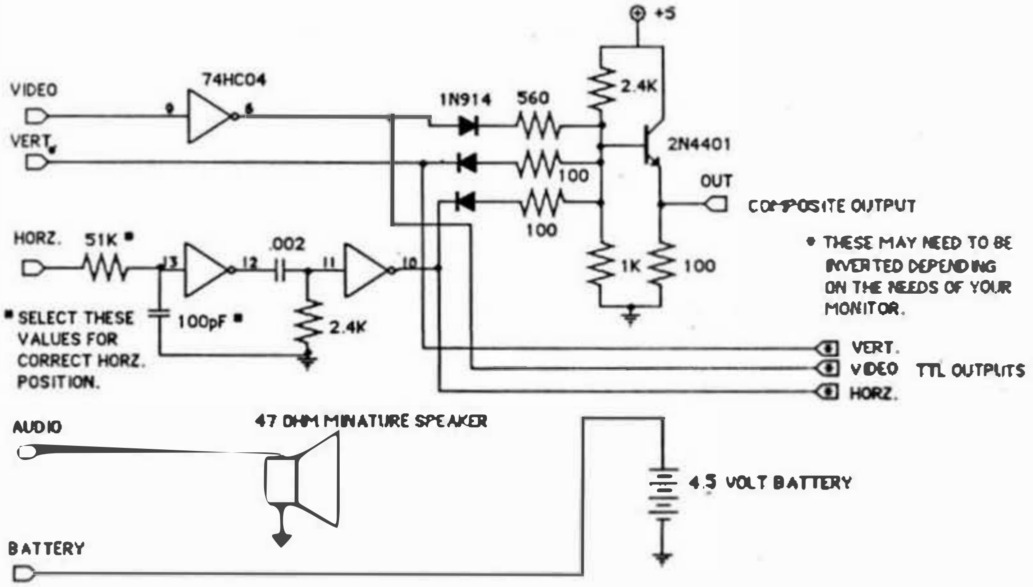
Mouse: Buy it for $100 from Apple, Get an aftermarket mouse like Mouse Systems A+ mouse ($60 at discount), Build a trackball like I described in a previous article, or whatever.

Box: Here's where you can go wild or mild. Just allow enough room for the board and power supply and disc drive. Try to keep the drive as far away from the power supply as possible and keep the drive cable as short as possible (<12"). Be sure to allow enough ventilation and access to the connectors on the back and a means of plugging the keyboard into the front of the board. The keyboard cable connector is a standard phone handset connector but the cable has a different end to end connection. try to get the Apple cord.

Monitor: The Mac puts out separate TTL level sync and video with a horzontal frequency of 22KHz. This is considerablly faster than TV standard of 15.7KHz. This poses some problems in the selection of monitors. However IBM's monochrome output is 18Khz which is usually close enough that a IBM compatible monitor can be adjusted to work. I have sucessfully used a Radio Shack VM-2 and with a minor modification a NEC 1201. I would guess that those El-Cheapo monitors based on TV parts might have a problem. Many places offer bare chassis monitors that are suitable, some for as little as $30. Size is not a problem. I use a 12". I have seen suitable 15" chassis advertised. At any rate you will have to build an adapter to match the monitor. The schematics for an example is included. Screen color is up to you. White matches the Mac, although I personally prefer amber. Green is my last choice, but personal preference and avaliability should decide for you.

If you shop around you should be able to come up with everything for <$500. Much less, if you already have some of the pieces. Attached are schematics for the hookup to the power supply, monitor and speaker.





Mouse Systems A+ Mouse: A review

The mouse that you got with your Mac is a reasonably functional unit. It does have some shortcomings. Being mechanical it is subject to wear and the pickup of dirt from the surface it's operated on.

The A+ mouse is an answer to some of these problems. It is a completly optical mouse which runs on a special grid pad. It won't wear out and isn't as sensitive to dirt as the standard mouse. It will only run on it's own pad which is a mixed blessing. The pad is slick and mouse action is smooth. The pad reserves mouse space on your desk which in my case at least is needed. However, it is a bit of a hassle sometimes to only be able to use it on the pad.

Basically I think it's a reasonable alternative to the original mouse. I wouldn't throw my original mouse away in favor of this one but if I needed to replace it I would consider this one. If your work area is subject to dirt and your mouse requires a lot of cleaning or if your work surface is unsuitable for the standard mouse this may be the answer. I like it.

Mike Wolf

SK*DOS

Preface:

*For the past several months we have been receiving calls and letters from readers, wanting information on the 68000 SK*DOS. Actually, it seems that Peter Stark, the author or SK*DOS has been so busy with this project that he had little time to write us a review.*

*I finally had to contact him and threaten to tell everyone that he had become a S100 user unless he got something in to me, for you, about SK*DOS. The following is the result.*

*Then Tom, over at Data-Comp saw this and told me that our new MUSTANG-08 is available with SK*DOS or OS-9. Well, how 'bout that?*

DMW

SK*DOS

SK*DOS 68K is a single-user disk operating system for computers using Motorola 32-bit CPUs such as the 68008, 68000, 68010, and 68020. It is ideal for applications in industrial control, scientific computing, turnkey systems, and just plain hobbyist use

It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes and more. Because SK*DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

At this time, SK*DOS / 68K is a strictly single-user, single-tasking operating system, but we anticipate that some users may want a multi-user or multi-tasking version in the future. Hence SK*DOS is designed to make future extensions easy.

The structure of SK*DOS / 68K is very similar to that used in our 6809 SK*DOS. It uses the same disk format, and so can read and write 6809 SK*DOS (and therefore also 6809 Flex) disks. Likewise, its appearance is the same, even the interface to user programs is very similar. Obviously, a 68000 DOS must be different from one designed for the 6809, but the differences have been handled in such a way that a user familiar with SK*DOS 6809 (or Flex) will feel right at home with SK*DOS / 68K.

We obviously would like SK*DOS / 68K to become widely used. For this reason we designed it with two criteria in mind.

First, SK*DOS / 68K should be affordable, so that even users of small or unpopular systems could have SK*DOS / 68K for their systems. Second, SK*DOS / 68K should be adaptable to a wide variety of systems, so it provides a common disk operating system which allows a program developed on one system to be run on many others.

SK*DOS /68K can be used in systems as small as 32K. We provide versions already configured for several systems; available now are versions for:

PT-68K 68008 computer from Peripheral Technology

ESB-I 68008 computer from Emerald Computers Inc

model 7950 68000 computer from NCR, and

MUSTANG-08 super micro from DATA-COMP Division of CPI.

We also provide a 'generic' version which is user-configurable for other systems (although a fairly knowledgeable user is a required element of this configuration.) Finally, we offer a licensing plan so that users who adapt SK*DOS / 68K to a new system may allow us to market their adaptations on a royalty basis.

An optional Configuration Manual, which describes how to implement SK*DOS / 68K on a new system, normally costs $50 but is being included at no charge with copies of SK*DOS / 68K sold until December 1st. It includes two disks with source code for the terminal and disk drivers, boot program, disk format utility, and other valuable information. Builders of small systems will be especially interested in the source code for 68K HUMBUG, a boot ROM which provides for system startup and debugging while adapting SK*DOS for new systems. The disks are available in either SK*DOS format (which can be read by 6809 SK*DOS or 6809 Flex), in Radio Shack Color Computer format, or in IBM PC format.

Our aim in writing SK*DOS has been to develop a powerful DOS, but one which is simple to use. Many prospective users have told us to "Keep It Simple, Star-Kits!" That is obviously not totally possible with a powerful CPU like the 68xxx, but SK*DOS / 68K is set up so that it defaults to a very simple system unless the user specifically calls its more advanced features. It is

1. Single-user and single-tasking (for now)
2. Non-relocatable. It always occupies a fixed location in memory.

3. Disk fonnat is compatible with 6809 SK*DOS. 68K SK*DOS can read 6809 SK*DOS disks, and vice versa, both for text and binary files (except for those 68xxx binary files with load or transfer addresses of 64K or above).

4. File Control Block structure is very similar to that of 6809 SK*DOS.

5. 68K SK*DOS functions are very similar to those of 6809 SK*DOS, and current 6809 SK*DOS (or Flex) users will find it an easy transition to the 68K SK*DOS.

6. Because 68K SK*DOS is used very much like 6809 SK*DOS, many existing programs can be translated on an almost line by line basis.

To run SK*DOS / 68K, *a computer requires a minimum of 32K of RAM*, with 128K or more being preferred. But the old maxim of 'the more, the better' certainly holds.

Although SK*DOS is not relocatable, we can provide it at a variety of different memory addresses. Since most 68000 computers will have RAM beginning at address $0000, the most likely memory configuration will look as follows:

$0000 - 03FC Trap and interrupt vectors $0400 - 0FFF Boot routines and stack space $1000 - 5FFF (approx) SK*DOS / 68K $6000 to end of memory - available to user programs

As indicated above, this is just one possible memory configuration, but the most likely one. Different configurations might be possible for systems which do not have available memory beginning at $0000.

All memory above SK*DOS / 68K is available for user programs and utilities. It is possible, however, to load some programs into memory and leave them there. For example, SK*DOS / 68K is supplied with a RAM disk program which can reserve from 16K to 1 megabyte of memory as a RAM disk. This program and its data area would normally be loaded once and then stay in memory above SK*DOS / 68K; SK*DOS / 68K maintains memory pointers which define the bottom and top of free memory.

Unlike SK*DOS / 68K itself, disk-resident commands (such as CAT or LIST), as well as user-written programs, are written in position independent code. When executed, these are loaded by SK*DOS / 68K into the free space above SK*DOS / 68K itself (and above any memory-resident programs that may already have been previously loaded.) Hence such user programs and utilities will run in any SK*DOS / 68K system, even one where SK*DOS / 68K has been located at some other address.

Since SK*DOS / 68K may not lie in the same place on all systems, user programs must be position independent. Moreover, all calls to SK*DOS / 68K, as well as all references to SK*DOS / 68K variables, must also be position independent. Hence all such calls are through exception vectors, and all references to SK*DOS / 68K variables are through relative addressing.

SK*DOS / 68K is provided with a full complement of utilities, including those needed to fonnat, copy, backup, and test disks: display disk contents; build, list, append, rename, or delete files; and change system parameters. This includes the following commands:

ACAT DELETE O SAVE
APPENDFORMAT OUTPIPE
SCAT BACKUPGETPEEK
SEQUENCE BEEP GETX POKE
*SK*DOS09 CAT INPIPE P*
STEPRATE COPY LINK
PARAMS SYSTEM DATE LIST
PROMPT TCAT FIND LOCATE
PROTECT VERSION BUILD
MAKEMPTY RAMDISK WORK
COMPARE MONRENAME XEQ
DAMON NOBEEP RESET

Some of these functions are quite useful, and available with most operating systems only at extra cost. For example, the RAMDISK command allows you to set up a RAM disk up to 1 megabyte in size, which behaves like any other drive.

A rather unique command is SK*DOS09. This is a complete 6809 SK*DOS and a 6809 emulator program. It allows you to run standard 6809 programs on your 68K system (although they do run at reduced speed). Until such time as true 68K versions become available, this allows you to run such popular programs as 6809 editors, spreadsheets, assemblers and cross-assemblers, word processors, compilers, and more.

Like its 6809 predecessor, SK*DOS / 68K contains program-callable functions to do file maintenance; open, read, write, and close sequential and random files; read and write individual sectors; access the disk directory; input and output characters, strings, and numbers; process command line arguments; parse file names; report errors, and more.

SK*DOS / 68K has also been designed to make expansion easy. A standardized system of calling disk and terminal drivers makes it possible to add additional device drivers without great complexity. SK*DOS / 68K will support up to ten logical disk drives, so a system can contain as little as one drive (plus perhaps a RAM disk), or as much as ten drives.

Files written to disk normally contain the file creation or last modification date. In addition, each file entry may also contain a creation or modification time if the system contains a clock / calendar IC. If such hardware is not available, then SK*DOS / 68K defaults to sequentially numbering files each day, and records the sequence number rather than the time. Disk catalog utilities are provided which list disk contents either in the order on the disk, in alphabetic order, in order by date and time, or in order by date and sequence number; the latter two show the latest files first.

As with any new DOS, availability of software is always a problem in the first few months. But a good selection of such software is either now available from third party vendors, or soon will be. In addition, 6809 editors or cross-assemblers may be run in emulation mode under SK*DOS. Finally, because its disk fonnat is identical with that of 6809 SK*DOS (and Flex), software development for SK*DOS is also easily accomplished on 6809 systems.

Development of third-party software to run under SK*DOS / 68K is under way. This is taking place in two areas:

1. Individual developers. Several initial SK*DOS / 68K users have developed software which will be available at either no cost or very low cost in the near future. Right now, a full screen editor, and a *"Small C" compiler are available for $10 each*

from John Byrns, 1953 Governors Lane. Hoffman Estates IL 60195. Other users are developing software such as word processors, utilities to read and write PC/MS-DOS disks, and higher level languages.

2. Commercial developers. *A 68000 assembler running under SK*DOS is now available for $50 from Computer Systems Consultants Inc., 1454 Latta Lane, Conyers GA 30207* (and this assembler will also be marketed by us.) In addition, a number of other firms have indicated that they plan to port their existing 6809 or 68xxx software to SK*DOS.

In order to make SK*DOS / 68K available as soon as possible, we have concentrated on basic functions first. There are several additional enhancements which will be provided as later (free) upgrades to initial purchasers. These include multiple directories, command line editing, enhanced input and output redirection, and a unified method of making screen and printer control totally transparent to user programs.

*In addition, SK*DOS / 68K has also been planned to allow simple future expansion to multi-user or multi-tasking operation. Development of such future versions will depend on demand.*

SK*DOS is available for single-copy or dealer sales, as well as OEM licensing. Single copies cost $125; extremely attractive OEM licensing terms are also available

SK*DOS (as well as other fine 68000 and 6809 hardware and software products) is available from:

Star-Kits Software Systems Corporation. P. O. Box 209. Mt. Kisco NY 10549. Telephone 914-241-0287, Telex 510 6016774. See advertising this issue.

Also Check with S.E. MEDIA Division - CPI. see catalog. this issue.

EOF

BIT-BUCKET

By: All of us..........................

Technical Systems Consultants, Inc.
111 Providence Rd.
Chapel Hill, North Carolina 27514

Dear Don,

I just received the August issue of "68 Micro" and was quite surprised to read Bud Pass's description of our 68020 C compiler. There were several inaccuracies about our compiler in his 'C User Notes' column. I would like to respond to these inaccuracies on a point by point basis.

First of all, it is important to note that we offer two C compilers. One for 68000/68010 systems, and one for 68020 systems. The 68020 compiler takes full advantage of the richer instruction set offered by that processor, as well as producing in-line floating-point instructions for the 68881. Floating point register declarations are also supported, making use of the 68881 register set.

Mr. Pass makes the statement that our compiler was "obviously derived from the 6809 McCosh C compiler." This statement is totally false. Our compiler was developed completely in-house by our own team of C compiler experts. About the only thing we have in common with the McCosh compiler is the fact that we accept the C language as input. I am not sure what criteria Mr. Pass based his conclusion on.

The next statement he makes is also false. Our compiler is described as "almost full C ... lacking only bit fields." Our compiler IS a full C compiler as per the System V specification. Nothing has been omitted. We support not only bit fields (which have been part of the C language for many, many years) but also the latest additions to C including enumerations, void data types, structure assignments, structure arguments to functions, and returned structures from functions. We also support unsigned chars, unsigned shorts, and unsigned longs as data types, along with the usual unsigned int.

Mr. Pass cites an example which he claims is incorrect in our compiler. The example he gives follows:

```
struct x1 {char c[10];};

struct x2 {char *c;};
```

He states that in both structures, the data is stored as a pointer to a string, therefore the 'sizeof' both structures is 4. This is definitely not the case. Structure x1 is 10 bytes long, whereas x2 is 4 bytes long (a pointer). If compiled on the 68020 version of our compiler, sizeof(x1) will yield 12 unless the 'Q' option is specified at compile time, in which case sizeof(x1) would yield 10 as expected. The extra two bytes in the first case are due to the compiler's automatic quad byte alignment of data elements which, by helping the cache in the 68020 improves execution times. The 'Q' option disables automatic quad byte alignment.

The next comment Mr. Pass makes is, "the compiler and linker are quite restrictive on the use of extern variables." Our compiler behaves identically to the System V compiler, but our linker does differ. The System V linker will allow multiple modules to contain the same global data definition (no 'extern' specifier in C). This encourages very bad programming practice and can often be the source of errors in a program. Our linker, like most non-UNIX linkers, does not allow this. We may ultimately add an option to the linker which would allow this, but that option would be added with great hesitancy on our part. As far as the C language is concerned, this ability is not required. Referencing page 206 of K&R:

The appearance of the 'extern' keyword in an external definition indicates that storage for the identifiers being declared will be allocated in another file. Thus in a multi-file program, an external data definition without the 'extern' specifier must appear in exactly one of the files. Any other files which wish to give an external definition for the identifier must include the 'extern' in the definition.

Even though most UNIX linkers do not make this restriction, we have decided to stick strictly to the C language specification.

By the way, the example which Mr. Pass gives in his column will compile, link, and run correctly on our system. To make it not link, you should remove the 'UNIV' from each line of the 'header.h' file. The way the sample is presented is correct according to K&R, and therefore compiles correctly on our compilers.

The last point Mr. Pass makes about our compiler concerns the 'printf()' function. He is correct that we do not accept 'pflinit', and it's totally by accident that we do accept 'pffinit'. These functions are completely unnecessary in our compilers, are not part of the System V standard, and are only found in McCosh derived compilers — far from a standard. Mr. Pass then goes on and presents an example which he says fails on our compiler:

```
printf("%05d", 5);
```

He is correct in that this will not cause leading zeros to be printed in the number. This format specification is that of older version 7 UNIX systems and the McCosh C compiler. As our documentation (as well as the System V printf document) states the correct format specification for leading zeros in this example is

```
printf("%d5.5", 5);
```

The System V 'printf' differs from previous versions, and we have adhered strictly to the System V definition.

One last comment about Mr. Pass's article. In his discussion of "Loss of Carrier", he mentions that the UniFLEX serial device drivers do not detect DCD on the port. This is not a problem with the drivers, but a lack of support by the hardware. The Mustang-020 does not support the DCD line on any port except port 4 (tty03). If a modem is to be used on this system, it should be connected to port 4 and DCD will act as expected.

To conclude, Don, we at Technical Systems Consultants enjoy seeing our products mentioned in your magazine, but we expect such reports to be accurate. Our compiler staff has put many man-years into the development of our C compiler and are quite proud of it. Not only has it been used to compile millions of lines of application programs from outside vendors, but it consistently passes a 1500-file (about 75,000 lines of code) proprietary C test suite from an outside source (Tektronix).

I hope this letter clears up any misconceptions created by Mr. Pass's article. If you or any of your readers have any questions concerning our C compiler, please do not hesitate to call us directly.

Sincerely,

David Shirk
President

Editor's Note: Thanks Dave for the letter. Below is Dr. Bud Pass's reply. Please note that I have not edited either, as to wording or content. I try to afford all an opportunity to reply to any letter we publish, or any article.

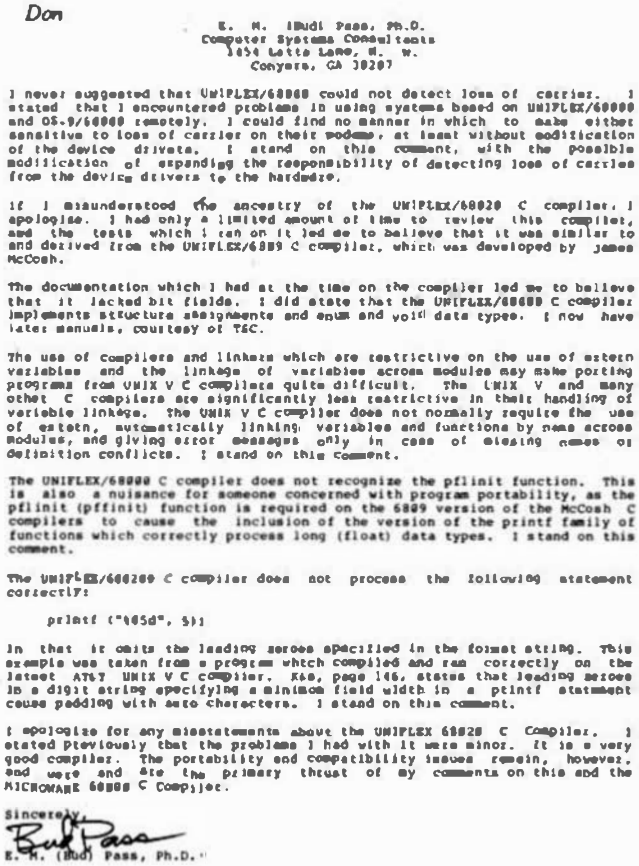
I recommend to all our readers, should they have any additional questions, they refer them directly to you (TSC) or to Dr. Pass. I have placed the TSC address at the top of this, so all those wishing additional information, from TSC, might write directly. Questions or comments to Dr. Pass should come directly to 68 MICRO JOURNAL or go to Dr. Pass. His address is included in each of his monthly articles.

You are correct on the MUSTANG-020 port 4 DCD support. This port was especially equipped with additional hardware to handle that type I/O.

I might add that I also was under the impression that the TSC C compiler was modeled after the McCosh 6809 series of C compilers. Especially with the inclusion of the 'pffinit' function.

Nice hearing from you Dave. Keep in touch.

Don



E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants
1454 Latta Lane, N. W.
Conyers, GA 30207

I never suggested that UNIFLEX/68000 could not detect loss of carrier. I stated that I encountered problems in using systems based on UNIFLEX/68000 and OS-9/68000 remotely. I could find no manner in which to make either sensitive to loss of carrier on their modems, at least without modification of the device drivers. I stand on this comment, with the possible modification of expanding the responsibility of detecting loss of carrier from the device drivers to the hardware.

If I misunderstood the ancestry of the UNIFLEX/68020 C compiler, I apologize. I had only a limited amount of time to review this compiler, and the tests which I ran on it led me to believe that it was similar to and derived from the UNIFLEX/6809 C compiler, which was developed by James McCosh.

The documentation which I had at the time on the compiler led me to believe that it lacked bit fields. I did state that the UNIFLEX/68000 C compiler implements structure assignments and enum and void data types. I now have later manuals, courtesy of TSC.

The use of compilers and linkers which are restrictive on the use of extern variables and the linkage of variables across modules may make porting programs from UNIX V C compilers quite difficult. The UNIX V and many other C compilers are significantly less restrictive in their handling of variable linkage. The UNIX V C compiler does not normally require the use of extern, automatically linking variables and functions by name across modules, and giving error messages only in case of missing names or definition conflicts. I stand on this comment.

The UNIFLEX/68000 C compiler does not recognize the pflinit function. This is also a nuisance for someone concerned with program portability, as the pflinit (pffinit) function is required on the 6809 version of the McCosh C compilers to cause the inclusion of the version of the printf family of functions which correctly process long (float) data types. I stand on this comment.

The UNIFLEX/680208 C compiler does not process the following statement correctly:

```
     printf ("%05d", 5);
```

In that it omits the leading zeroes specified in the format string. This example was taken from a program which compiled and ran correctly on the latest AT&T UNIX V C compiler. K&R, page 146, states that leading zeroes in a digit string specifying a minimum field width in a printf statement cause padding with zero characters. I stand on this comment.

I apologize for any misstatements about the UNIFLEX 68020 C Compiler. I stated previously that the problems I had with it were minor. It is a very good compiler. The portability and compatibility issues remain, however, and were and are the primary thrust of my comments on this and the MICROWARE 68000 C Compiler.

Sincerely,

Bud Pass

E. M. (Bud) Pass, Ph.D.

Note to all vendors (advertisers or not): It takes several months to get something submitted by one of our regular authors from disk to print. And we are one of the 'short time' magazines. Most take months longer than we do. Note the dates of letters in Bit Bucket. So, if vendors of products that are the subject from time to time, of articles appearing in 68 MICRO JOURNAL, would keep this in mind and if they do desire to keep the authors (and specs) current, they should take into account our time lag.

It seems that some vendors only update our staff when we run into a misunderstanding. It would benefit us all if we could be kept updated more promptly. Sometimes we receive information and specifications, about some product, months after it has gone to market.

I know it pays also. Many of those vendors who have kept us appraised over the years are still going strong. We help sell products, but only when we have good information.

If vendors do not see fit to keep our staff current, we can only go on what we have on hand! We would much rather be right the first time than to have to retell part of it over again!

DMW

10912 Knights Bridge Ct.
Reston Virginia 22090
August 29, 1986

Don:

I should have written this several weeks ago but I kept putting it off because of class work for my MBA courses. Now that I'm pretty well caught up (at least I have that to be thankful for), I have no more excuses. I did a recent up-grade that I think some of your readers might be interested in. Because the price of 256k DRAMS has come down dramatically in the last few months, and because I was running out of virtual disk space, I decided it was time to increase the capacity of my Computer Excellence Memory board. I decided I couldn't afford the whole 32 chip set yet and I really didn't need a whole megabyte anyway, so I bought sixteen of the chips from my local computer store. I studied the diagrams. I thought I had it figured out. I plugged in the chips, changed the jumpers around and tried the board. Nothing. It's the only memory board in the system. The computer would't boot.

Now I figured I was in trouble. I calmly put the old 64's back in. To my horror -- still nothing! Now I'm in real trouble. I plug in my trusty GIMIX 16K static board. Thank heaven, the disk drive starts to dance the OS-9-boot-polka. Now I knew the trouble was with the memory board itself. There was nothing for it but send it back to Computer Excellence for repair. At least I had the forsight to finish my homework for school before I started the project. I would need it next week. Could I get it back in time to get my next assignment done? I called Computer Excellence to find out how long it would take to fix the board. To my surprise, after Malcomb heard about my problem, he said it might not be bad enough to send in. If I had a 'scope handy, we might find out what was wrong with it over the phone. An hour later I found a memory buss driver chip with one leg that had never been soldered to the board. A touch with the iron and I had 256K again. Malcomb even waited on the phone while I plugged the 256's back in and set the strapping to give me 640K (16 256's plus 16 64's). The $25 long distance phone bill was the only cost for this "service call.

The upgrade itself looks like it may be more complicated than it really is. I thought I might pass on a few tips. First the most economical partial upgrade is replacing half the 64K chips on the board with 256's. After buying the chips, the addressing on the board must be set to use the 256K chips. This means jumpers JS3,4,9,10 and JS12 must be installed as outlined on pages 2-1 and 2-2 of the Computer Excellence manual. JS3,4,9, and JS10 will each have pin "a" connected to pin "b" and pin "d" connected to pin "e." Finally, jumper JS12 has pin "c" connected to pin "f." This is the setup even though half the chips are 64K! It is also necessary to have switch S4 in the OPEN position if it is not all ready so.

Now for the hard part. Very carefully remove the LOWER (ie. the end closest to the 50 pin buss connecter) two rows of 64K chips. This is bank 0 and 1. Replace these with your 16 256K chips. If all has gone well, you should have a board which can address 1024K. Wait a minute! I know I said 640K before. The smaller number is the correct one. If you experiment a little, you find that the top 384K are "ghost" images of the next lower 128K. What I am saying is the bottom 512K is from the two banks of 256K chips. The sixteen 64K chips provide 128K above that. The addresses higher than 640K (HEX A0000-FFFFF) just repeat the data from the 64's. I hope my experiences with the upgrade will be of help to some one else.

Best regards,

OE Groves

Editor's Note: Thanks OE for the letter and info. We all know what a sad feeling it is when you get the sudden impression that something you did, blew the whole system.

I hope the information helps some of the others, there are a lot of Computer Excellence boards out there. Don't have one personally, but all I hear is good things about their products.

You might look to some new and current technology from them soon. Last time I talked to Malcomb, he was bubbling over about somethings they were into that will interest us all.

Thanks for the report on the excellent service from COMPUTER EXCELLENCE, it is folks like Malcomb that has kept us going.

DMW

CMDRPT

Command Line Recall and Modification for FLEX

Dave Parr, Coventry Polytechnic,
Priory Street, Coventry,
England CV1 5FB

I have always enjoyed using the FLEX operating system on a variety of hardware, but I have often been irritated by the lack of one particular feature, namely the ability to repeat the previous command by a single keystroke.

Many is the time I have typed in a long command line, executed it, then found I wanted to do exactly the same thing again and had to re-type the entire line; alternatively I have entered a long command and got some sort of error back from FLEX, frequently because of a typing mistake or a failure to specify the correct drive number for a filename.

What I needed was a way to recall the previous command line to the screen and then either re-execute it immediately or make minor modifications and then execute it.

I had seen utilities of the type

```
REPEAT flexcommand <CR>
```

which would prompt REPEAT(Y or N)? after execution of 'flexcommand', but while satisfying my first requirement above, modification of 'flexcommand' was not possible . Anyway it would be a bit tedious to prefix every command with REPEAT on the off-chance that the odd one would need repeating after executing once.

As a result, CMDRPT was born out of my frustration, and the source code is shown in the listing which follows. It should be assembled to a .BIN file on the system disk and the command GET 0.CMDRPT should be included in the STARTUP file.

As you will see from the source, it is possible to change the single byte location RPTCH to make any non-printing key recall the previous command line. I have used CONTROL A on SWTPc systems, the BREAK key on a Dragon 64 CoCo, and various others where convenient. Any function key could be used where available.

OPERATION

When the character chosen for RPTCH is entered, the previous command line is immediately displayed on the screen, with the cursor at the end of the line, as if it had just been typed.

If a <CR> is now entered, the command will be executed again as you would expect; if instead more characters are typed, these will be added to the command line on the screen and a <CR> will then start execution of the amended command line. For example:

```
LIST FILENAME.OUT
could be changed to
LIST FILENAME.OUT.0

P ASMB FILE1 FILE1.CMD
could be changed to
P ASMB FILE1 FILE1.CMD +Y
```

If backspacing is used, it is possible to modify the line (or usefully the latter part of it) and thus correct typing mistakes. For example:

```
DELETE 1.FILENAME.TZT
could be changed to
DELETE 1.FILENAME.TXT
```

Furthermore, since the repeat key can be depressed at any time during command line input, it is possible to add characters to the front of the previous command line as well as at the end. I find this very useful when a test assembly of a file turns out to be error-free and I then want a printed listing and a binary output file. For example:

```
ASMB FILENAME +BLS
could be preceded by "P,"
then recalled, giving
P,ASMB FILENAME +BLS
then entering three backspaces
and typing "GY" would give
P,ASMB FILENAME +GY
```

Obviously, the longer the command line, the more time and effort are saved. I personally find that avoiding the need to re-type a string of characters I have just entered brings satisfaction out of all proportion to the work involved in re-typing them !

As a final suggestion for more exotic use, a long command which contains repeated occurrences of the same piece of text can be entered more easily by first typing this text as a command, ignoring the WHAT? or NOT FOUND response, then using the comand repeat key to insert this text into the new command line at the appropriate places. For instance, this can be useful to avoid multiple typing of an awkward filename which is required to appear several times in a command line.

The number of times that the previous command can be recalled is limited only by the size of the line buffer; when 127 characters are present on the command line, further recalls are ineffective.

THEORY

Both the WARMS code and the INBUFF code in FLEX are patched. The new INBUFF code sets the carry flag if the command repeat key is pressed, and the new WARMS code detects this after each return from INBUFF. Each time a command is issued, it is copied from FLEX's line buffer to a second buffer of the same size.

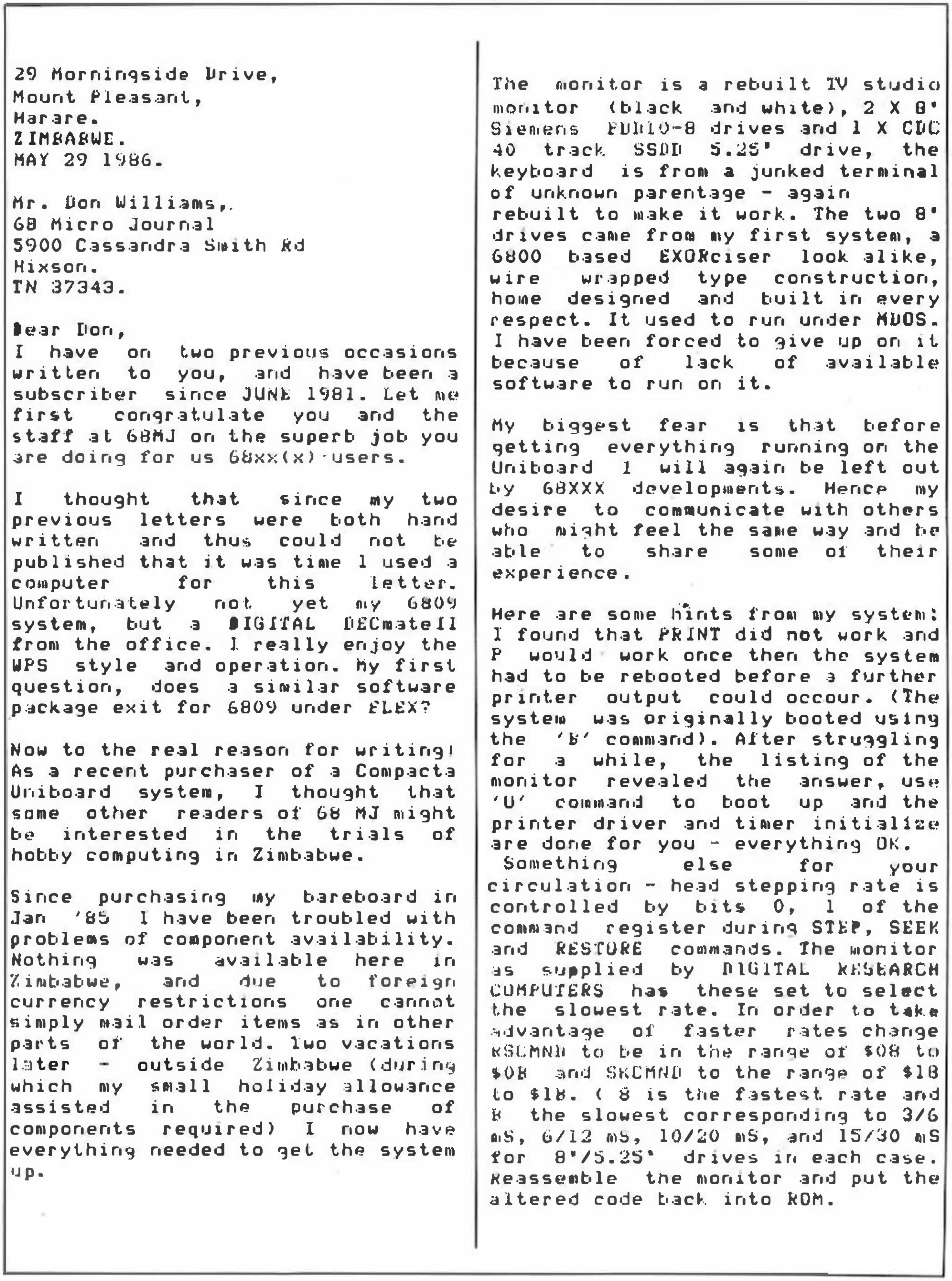
If the command repeat key is pressed at any time while entering characters via INBUFF, the contents of this second buffer are copied into the line buffer, after any characters which may already have been put there. The INBUFF routine is then re-entered to allow further input.

The main code requires 96 bytes and the new buffer requires 128 bytes; these two areas can be separated in memory if desired.

Any areas within FLEX which are free on a particular system can be used e.g. spooler, console or disk drivers etc.

```
                    **************************************************************
                    *                                                            *
                    *       CMDRPT                                               *
                    *                                                            *
                    *       Allows single-key command line recall and modification
                    *                                                            *
                    *       May be invoked by GET 0.CMDRPT in STARTUP file       *
                    *                                                            *
                    *       Dave Parr                                            *
                    *       Coventry Polytechnic                                 *
                    *       Coventry  CV1 5FB                                    *
                    *       England                                              *
                    *                                                            *
                    *       16 July 1986                                         *
                    *                                                            *
                    **************************************************************


        C080 LINBUF EQU   $C080   FLEX line buffer
        CC00 TTYBS  EQU   $CC00   backspace character
        CC14 LBPTR  EQU   $CC14   line buffer pointer
        CD81 RENTRY EQU   $CD81   start of code for RENTER routine
        CE2B INBUF  EQU   $CE2B   start of code for INBUFF routine
        CE31 INBUFA EQU   $CE31   location of JSR GETCH in INBUFF code
        CE3C INBUFB EQU   $CE3C   instruction after CMPA TTYBS in INBUFF code
        CE55 INBUFR EQU   $CE55   location of RTS at end of INBUFF code
        CEB1 PSTRN  EQU   $CEB1   start of code for PSTRNG routine
        CEB3 PSTRNA EQU   $CEB3   entry point to avoid CRLF in PSTRNG code

        0004 EOS    EQU   $04
        000D CR     EQU   $0D

                    *
                    * jump from end of 'warm start' code so
                    * can respond if the 'command repeat
                    * character' is detected by 'inbuff'
                    *
CDAC                       ORG   $CDAC

CDAC 7E   C735             JMP   WARMS2   overwrites JSR PSTRN

                    *
                    * jump from 'inbuff' code after checking for
                    * TTYDEL character, to test whether the
                    * 'command repeat character' has been input
                    *
CE39                       ORG   $CE39

CE39 7E   C723             JMP   INBUF2   overwrites CMPA TTYBS

                    *
                    * extra code requires 96 bytes; ORG it to any suitable hole
                    *
C720                       ORG   $C720    *** spare in my (non-spooling) system ***

C720 01             RPTCH  FCB   $01      *** choose your own non-printing key ***
C721                SAVPOS RMB   2        saves position in LINBUF where copy starts
```

```
                    *
                    * patch to 'inbuff' to check if the 'command
                    * repeat character' has been input
                    *
                    * if so, exits with carry set, otherwise clears carry
                    *
      C723 B1   C720     INBUF2 CMPA  RPTCH    command repeat char input ?
      C726 26   05              BNE   NOTRPT   no
      C728 1A   01              SEC            yes, set carry ...
      C72A 7E   CE55            JMP   INBUFR   ... and go to RTS at end of 'inbuff'
      C72D B1   CC00     NOTRPT CMPA  TTYBS    replace overwritten instruction
      C730 1C   FE              CLC            clear carry
      C732 7E   CE3C            JMP   INBUFB   return to 'inbuff' where left off

                  *
                  * patch to 'warms' to respond to detection of
                  * the 'command repeat character' by 'inbuff'
                  *
                  * if not detected, copy the newly-entered command line from
                  * FLEX's line buffer to the save buffer
                  *
                  * if detected, add the contents of the save buffer to
                  * any characters already entered into the line buffer
                  * via 'inbuff', then call 'inbuff' again to
                  * allow further input
                  *
    C735 BD   CEB1     WARMS2 JSR   PSTRN    replace overwritten instruction ...
    C738 BD   CE2B            JSR   INBUF    ... and the next one not executed
    C73B 25   12       CMDRPT BCS   REPEAT   carry set so go do repeat code
    C73D 108E C080     NORMIN LDY   #LINBUF  normal input, so copy LINBUF to SAVBUF
    C741 8E   C780            LDX   #SAVBUF
    C744 A6   A0       CPYBYT LDA   0,Y+
    C746 A7   80              STA   0,X+
    C748 81   0D              CMPA  #CR
    C74A 26   F8              BNE   CPYBYT
    C74C 7E   CD81            JMP   RENTRY   return where left off at 'renter' code
    C74F 108E C780     REPEAT LDY   #SAVBUF  prepare to copy SAVBUF to LINBUF
    C753 BF   C721            STX   SAVPOS   record next position in LINBUF
    C756 A6   A0       GETBYT LDA   0,Y+     get a byte from SAVBUF
    C758 81   0D              CMPA  #CR      is it CR ?
    C75A 27   05              BEQ   PUTBYT   if CR, go copy it
    C75C 8C   C0FF            CPX   #LINBUF+127 if not CR, check for end of LINBUF
    C75F 27   06              BEQ   TERMIN   if at end, go terminate it for PSTRNG
    C761 A7   80       PUTBYT STA   0,X+     copy a byte to LINBUF
    C763 81   0D              CMPA  #CR      was it CR ?
    C765 26   EF              BNE   GETBYT   if not CR, get another byte
    C767 86   04       TERMIN LDA   #EOS     if CR or end of LINBUF, use EOS...
    C769 A7   82              STA   0,-X     ... to overwrite last char added
    C76B 34   10              PSHS  X        save current position in LINBUF
    C76D BE   C721            LDX   SAVPOS   point at copied bytes
    C770 BD   CEB3            JSR   PSTRNA   print them without CRLF
    C773 8E   C080            LDX   #LINBUF  set start of line buffer ...
    C776 BF   CC14            STX   LBPTR    ... into line buffer pointer
    C779 35   10              PULS  X        restore pointer after last copied byte
    C77B BD   CE31            JSR   INBUFA   re-enter at JSR GETCH for further input
    C77E 20   BB              BRA   CMDRPT   after 'inbuff', go check again for RPTCH

                  *
                  * these 128 bytes of storage may be placed elsewhere in
                  * a separate spare area of memory
                  *
    C780 0D            SAVBUF FCB   $0D      in case copy after STARTUP
    C781                      RMB   127

                              END

0 ERROR(S) DETECTED

          SYMBOL TABLE:

          CMDRPT C73B   CPYBYT C744   CR     000D   EOS    0004   GETBYT C756
          INBUF  CE2B   INBUF2 C723   INBUFA CE31   INBUFB CE3C   INBUFR CE55
          LBPTR  CC14   LINBUF C080   NORMIN C73D   NOTRPT C72D   PSTRN  CEB1
          PSTRNA CEB3   PUTBYT C761   RENTRY CD81   REPEAT C74F   RPTCH  C720
          SAVBUF C780   SAVPOS C721   TERMIN C767   TTYBS  CC00   WARMS2 C735
```



29 Morningside Drive,
Mount Pleasant,
Harare.
ZIMBABWE.
MAY 29 1986.

Mr. Don Williams,
68 Micro Journal
5900 Cassandra Smith Rd
Hixson.
TN 37343.

Dear Don,
I have on two previous occasions written to you, and have been a subscriber since JUNE 1981. Let me first congratulate you and the staff at 68MJ on the superb job you are doing for us 68xx(x) users.

I thought that since my two previous letters were both hand written and thus could not be published that it was time I used a computer for this letter. Unfortunately not yet my 6809 system, but a DIGITAL DECmateII from the office. I really enjoy the WPS style and operation. My first question, does a similar software package exit for 6809 under FLEX?

Now to the real reason for writing! As a recent purchaser of a Compacta Uniboard system, I thought that some other readers of 68 MJ might be interested in the trials of hobby computing in Zimbabwe.

Since purchasing my bareboard in Jan '85 I have been troubled with problems of component availability. Nothing was available here in Zimbabwe, and due to foreign currency restrictions one cannot simply mail order items as in other parts of the world. Two vacations later - outside Zimbabwe (during which my small holiday allowance assisted in the purchase of components required) I now have everything needed to get the system up.

The monitor is a rebuilt TV studio monitor (black and white), 2 X 8" Siemens FDD10-8 drives and 1 X CDC 40 track SSDD 5.25" drive, the keyboard is from a junked terminal of unknown parentage - again rebuilt to make it work. The two 8" drives came from my first system, a 6800 based EXORciser look alike, wire wrapped type construction, home designed and built in every respect. It used to run under MDOS. I have been forced to give up on it because of lack of available software to run on it.

My biggest fear is that before getting everything running on the Uniboard I will again be left out by 68XXX developments. Hence my desire to communicate with others who might feel the same way and be able to share some of their experience.

Here are some hints from my system: I found that PRINT did not work and P would work once then the system had to be rebooted before a further printer output could occour. (The system was originally booted using the 'B' command). After struggling for a while, the listing of the monitor revealed the answer, use 'U' command to boot up and the printer driver and timer initialize are done for you - everything OK.
Something else for your circulation - head stepping rate is controlled by bits 0, 1 of the command register during STEP, SEEK and RESTORE commands. The monitor as supplied by DIGITAL RESEARCH COMPUTERS has these set to select the slowest rate. In order to take advantage of faster rates change RSCMND to be in the range of $08 to $0B and SKCMND to the range of $18 to $1B. (8 is the fastest rate and B the slowest corresponding to 3/6 mS, 6/12 mS, 10/20 mS, and 15/30 mS for 8"/5.25" drives in each case. Reassemble the monitor and put the altered code back into ROM.

Help please:
NEWDISKS as supplied with my system, bombs out so I cannot use my little drive! Eric Sillanpaa's article reveals that changing R33/C41 will sort out double density problems with 5.25''s but will my 8''s still work or do I' have a single size system. (I just have written directly to Eric about this.) What is the cure?

When I have everything going and can obtain software like XDMS, All-in-1, Stylograph, Dynacalc etc, will they all work or do I have the same problem as with my 6800 system? I have reworked many packages in the past, and really don't want to have to keep doing this ad-infinitum!

I have also written to John Cooper (see 68 MJ July '84 p48) re formation of a user group for Uniboard systems. If anyone else feels inclined to contact me I will attempt to respond.

Keep up the good work I look forward to receiving each issue.

Yours sincerely,

Preston

Neil Preston.

Editor's Note: Thanks for the kind words about 68 MICRO JOURNAL. That always makes our day!

Sorry, but I cannot give you any reply as concerns the DECmate II. Maybe some reader can drop us or you a line, with the information you desire. However, there are several different word processors advertised in the S.E. MEDIA catalog (center section) that should run quite well on the Compacta system.

As to support for that system, it seems to be there. As you mentioned, John Cooper may be able to help. We have had several articles and 'BIT BUCKET' shorts, that touch on the subject. Once all sorted out, it appears to be a nice, low priced system. We have one, and it does a nice job running FLEX. We have not tried OS-9 on ours. Mainly because we have about 6 other OS-9 systems in the office.

If you use the S.E. MEDIA version of FLEX, I do not see why most all the software you mentioned will not run. I guess if I am wrong, we both will hear about it soon. Thats the nice thing about our crowd. They let you know!

Hope this get you some help Neil. Keep in touch.

DMW

Reminder text file - see source.

```
1 1 ** NEW YEAR'S DAY
2 8 ** ART WELLER'S BIRTHDAY (SEND GIFTS!)
8 13 ** DON WILLIAMS BIRTHDAY (ME ALSO!!)
2 12 ** LINCOLN'S BIRTHDAY
2 15 ** WASHINGTON'S BIRTHDAY
7 4 ** INDEPENDENCE DAY
9 1 86 LABOR DAY
10 13 86 COLUMBUS DAY
11 11 ** VETERAN'S DAY
11 17 86 THANKSGIVING DAY
12 25 ** CHRISTMAS DAY
```

+++

```
/* REMINDER COMMAND -- READS A REMINDER FILE */
/* Art Weller, Jul 1986, Rev 2 */

   /* REMINDER.CMD is recommended for use with a STARTUP
file. REMINDER.CMD searches drive 0 for a file named
REMINDER.TXT containing the data on which it works. The TXT
file may be prepared and maintained with any TSC compatible
editor, but must be in the format:

  MM DD YY (text information on one line)

   The date is compared with the system date and if equal to
that and less than the added "look-ahead" days, the data is
listed to the CRT. Leading spaces are allowed in the dates.

   The datum, YY, is treated differently in that a
"who-cares" field is allowed by substituting "**" for the YY
entry:

  7 4 ** Independence Day -- a holiday.

   Would give you a reminder each year, in advance of the
holiday by the "look-ahead" period. "Look-ahead" has been set
arbitrarily to 15 days in the PROCEDURE GET_SYSDATE; may be
changed and re-compiled.

   Obviously, birthdays, anniversaries and the like are
candidates for this system and I've also found it useful to
remind of auto inspections, routine maintenance, etc.

   I recommend the file entries be made in "month, day"
sequence as it makes it easier to maintain, but this is not a
necessity. */


ORIGIN=$A000;
STACK=*;

GLOBAL BYTE    MATCH, CHAR;
     INTEGER MTH, MTH1, DAY, DAY1, YR, YR1, PTR;
     REAL    LODATE, HIDATE;

INCLUDE FLEX.LIB;
INCLUDE IOSUBS.LIB;

CONSTANT TRUE=-1, FALSE=0;

AT $C080 BYTE LINBUF;
AT $C840 BYTE INFILE, ERROR;
AT $CC0E BYTE SYSMTH, SYSDAY, SYSYR;
AT $CC14 INTEGER BUFPTR;
```

```
PROCEDURE CHK_ERROR(BYTE .FCB);
  IF ERROR AND ERROR <> 8 THEN
  BEGIN
    REPORT_ERROR(.FCB);
    CLOSE_FILE(.FCB);
    FLEX;
  END;
ENDPROC;

PROCEDURE GET_SYSDATE;
MTH = SYSMTH; MTH1 = SYSMTH;
DAY = SYSDAY-1; DAY1 = DAY + 15;
YR = SYSYR; YR1 = SYSYR;

  IF DAY1 > 30
  THEN BEGIN
    DAY1 = DAY1 - 30;
    MTH1 = MTH1 + 1;
  END;
  IFMTH1 > 12
  THEN BEGIN
    MTH1 = 1;
    YR1 = YR1 + 1;
  END;

  LODATE = (YR * 10000) + (MTH * 100) + DAY;

  HIDATE = (YR1 * 10000) + (MTH1 * 100) + DAY1;

ENDPROC;

PROCEDURE GET_A_LINE: BYTE I, CHAR;
 I=0;
 WHILE I<127
   BEGIN
   CHAR=READ(.INFILE);
   LINBUF(I)=CHAR;
   IF LINBUF(I)=$0D THEN BREAK;
   I=I+1;
   END;
 LINBUF(I)=0;
ENDPROC;

PROCEDURE NEXTNUM(BYTE .BUFFER);
 INTEGERN:
 BYTE CHAR, DONE;
 N =0;
 DONE = FALSE;
 WHILEBUFFER(PTR) = SP PTR = PTR + 1;
 REPEAT
   IF BUFFER(PTR) = '-'
     THEN BEGIN
      N=$FF;
       WHILE BUFFER(PTR) = '-' PTR= PTR+1;
      RETURN N;
     END;
   CHAR = BUFFER(PTR) - '0'; /* CONVERT FROM ASCII */
   IF CHAR < 0 .OR CHAR > 9 THEN DONE = TRUE;
   IF DONE = FALSE THEN
   BEGIN
     N = N * 10 + CHAR;
     PTR = PTR + 1;
   END;
 UNTIL DONE;
ENDPROC N;

PROCEDURE CHECK_DATES: INTEGER FMTH, FDAY, FYR;
            REAL   THISDATE;
 PTR=0;
 MATCH=0;

 FMTH = NEXTNUM(.LINBUF);
  IF FMTH <> MTH AND FMTH <> MTH1 THEN RETURN;
 FDAY = NEXTNUM(.LINBUF);
 FYR = NEXTNUM(.LINBUF);
 IF FYR < 0 AND YR = YR1 THEN FYR = YR;

 IF FYR > 0
  THEN BEGIN
  THISDATE = (FYR * 10000) + (FMTH * 100) + FDAY;
  IF THISDATE < LODATE .OR THISDATE > HIDATE THEN RETURN;
 END;
 ELSE BEGIN
   THISDATE = (YR1 * 10000) + (FMTH * 100) + FDAY;
   IF THISDATE < LODATE .OR THISDATE > HIDATE THEN RETURN;
 END;

 MATCH = 1;
ENDPROC;

PROCEDURE MAIN;
 BUFPTR="0.REMINDER.TXT";
 GET_FILENAME(.INFILE);
 SET_EXTENSION(.INFILE,1);
 OPEN_FOR_READ(.INFILE);
 CHK_ERROR(.INFILE);
 GET_SYSDATE;
 CRLF; CRLF;
 REPEAT
   GET_A_LINE;
   CHECK_DATES;
   IF MATCH THEN
     BEGIN
       PRINT(.LINBUF); CRLF;
     END;
 UNTIL ERROR;
 CHK_ERROR(.INFILE);

/* END REMINDER */
```

+++

Company Profile
of
TEC

P.O.Box 53 West Glover, Vermont 05875
Phone (802) 525-3458

TEC designs, manufactures and markets a line of microcomputer development tools that run on IBM and compatible personal computers. TEC's products are designed with the philsophy that as many of the development functions as possible should be performed by the personal computer. This philosophy minimizes the amount of dedicated hardware and software required and therefore minimizes development system cost. Currently, TEC offers development systems for the Motorola 6805 line of single chip microcomputers. These systems include a cross assembler program, a full screen simulator/debugger program and programming circuit boards.

TEC stands for The Engineers Collaborative and was founded by Robert B. Johnson who holds a Bachelor of Science Degree in Electrical Engineering from the University of Vermont. His prior experience includes integrated circuit design with the IBM Corporation and just before starting TEC he held the position of Director of Engineering at Powerline Communications Inc. in Burlington, Vermont.

New products currently under development at TEC include microcomputer development tools for the Hitachi 6305 series and Motorola 68HC11 series single chip microcomputers.

Information on TEC's products can be obtained by contacting TEC at the above address or by calling (802) 525-3458.

GESPAC

LOW COST INTERFACE BOARD GIVES FOUR SERIAL COMMUNICATION PORTS TO THE G-64 BUS

MESA, AZ, August 8, 1986--GESPAC introduces an inexpensive 4 channel serial communication board. The GESSIO-2 is built on a single height Eurocard and is compatible with the standard G-64 bus.

The G-64 bus is an easy-to-interface, non-multiplexed, 16-bit bus aimed at mid-range industrial applications.

The GESSIO-2 provides the system integrator with four programmable UARTs which allows a G-64 bus microcomputer system to communicate with a variety of devices such as modems, data terminal, and other devices equipped with a serial communication port.

The board uses four independent MC 6850 Asynchronous Communication Interface Adapter devices. An on board crystal oscillator allows the user to program the bit rate for each communication port for a variety of speeds ranging from 50 to 19,000 bauds. The board is equipped with the necessary transceivers and connectors to make it fully RS-232 compatible.

The GESSIO-2 is upward compatible with GESPAC's GESSIO-1E and GESSIO-1A, which are functionally similar but equipped with two channels. The GESSIO-2 will effectively replace two of these earlier boards without the need for modification in the software.

The GESSIO-2 can be used with any of the 8 and 16-bit microprocessors available on the bus, such as the Z80, 6809, 68000, and 80286.

The GESSIO-2 is available today at the low unit price of $375.

For more information contact:

Joe Murphy
GESPAC, Inc.
100 W. Hoover Ave.
Mesa, AZ 85202
(602) 962-5559

MOTOROLA
Semiconductor Products Inc.
P.O. BOX 20912 PHOENIX, ARIZONA 85036

Innovative systems through silicon.

EDITORIAL CONTACT:
Ed Prestwood
(602) 952-3610

INQUIRY RESPONSE:
Mr. R.Q. Green
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA REDUCES VMEbus PROCESSOR AND RAM BOARD PRICES BY UP TO 25%

Phoenix, AZ, August 18, 1986... Significant price reductions on 15 selected VMEbus compatible processor and RAM boards have been announced by Motorola's Microsystems operation. As a result of manufacturing economies, price reductions of up to 25% are now in effect.

According to Ron Bishop, Marketing Manager for Motorola's Microsystems products, "Independent market share estimates indicate that Motorola is the #1 worldwide supplier of VMEbus products. We feel we have attained this position through the breadth of our VME product line, by offering high quality at reasonable cost, and by providing good application support and customer service. These price reductions reinforce our commitment to be the price leader among broadline VME vendors."

The reduced prices shown in the attached table, are in U.S. dollars for U.S. delivery only. These VMEmodule™ products are available now through Motorola Semiconductor Sales Offices nationwide, plus authorized systems product distributors and representatives.

August 11, 1986

68 Micro Journal
Mr. Don Williams
Computer Publishing, Inc.
P.O. Box 849
Hixson, Tennessee 37343

Dear Don,

Thank you for sending us a copy of your magazine's review of CSG IMS. We feel that the information stated in the review is, in general, quite accurate although there are one or two trivial points which only experienced CSG IMS users would question.

One point that needs clarification, however, is that the English Language Query for CSG IMS on OS9/68000 is definitely a future product. We have not, at this point, even started to design this product and therefore do not anticipate releasing it any sooner than the fourth quarter of 1987.

I have also enclosed a news release reflecting the current status of CSG IMS. You may wish to include it in the next issue of 68 Micro Journal.

Thank you,

Edward Kebler

Clearbrook Software Group Inc.
(604) 853-9118
U.S.: 448 Harrison Street, P.O. Box 8000-499, Sumas, WA 98295-8000

microware
MICROWARE SYSTEMS CORPORATION

NEWS RELEASE

Microware Systems Corporation
Andrew (Drew) Crane
515-224-1929

OS-9 SELECTED FOR EUROPEAN STANDARD FOR EDUCATIONAL COMPUTERS

FOR IMMEDIATE RELEASE

Des Moines -- Microware has announced that Thomson SIMIV of Paris, France has licensed the OS-9/68000 Operating System for the software nucleus of the recently announced European Education Standard microcomputer system.

Three major European electronics companies including Thomson, the French electronics and defense group, Ing. C. Olivetti of Italy and Acorn Computer Group PLC, the UK educational and home computer company, have previously signed an agreement to cooperate in the development of a European Standard for 16 bit microcomputers incorporating OS-9/68000.

The three companies have agreed to join forces to develop a new personal computer primarily for the education sector and for the general home market and a new European standard for personal computers.

OS-9 SELECTED FOR EUROPEAN STANDARD...

By setting up a joint effort, the three partners intend to develop an internationally competitive base of hardware and software with special emphasis on the educational market. Under the agreement, Thomson's consumer electronics division, Olivetti and Acorn Computer Group, in which Olivetti has an 80% interest, will work to develop integrated software and hardware. The agreement offers technological advantages because it enables individual research and development to be directed toward a common goal.

Details regarding the features and specifications of the new system will be announced in the near future by Thomson and Olivetti. The product is expected to be introduced in early 1987.

Thomson, the largest manufacturer of home computers in continental Europe, has made a major effort in the educational sector of the personal computer market and is currently supplying the bulk of the 120,000 educational computers purchased this year by the French government for local schools.

OS-9 SELECTED FOR EUROPEAN STANDARD...

This announcement is the latest in a series reflecting OS-9's rapidly increasing acceptance as the standard operating system for 68000 based personal computers. Earlier this year Philips and Sony announced OS-9/68000 will be the basis for the Compact Disc - Interactive (CD-I) standard. In late July, Tandy released the Color Computer III based on Microware's operating system and graphics user interface.

OS-9 is a modular, real-time, multi-tasking operating system for computers based on the 68000 family of microprocessors. It is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1983, OS-9/68000 has been licensed to over 250 manufacturers for use in a wide variety of industrial, scientific and consumer products. Some of these products include personal computers, industrial control systems, data processing equipment and telecommunications systems.

Founded in 1977, Microware specializes in the development of advanced 68000 family operating systems and programming languages. Microware's offices are located in Des Moines, Iowa and Tokyo, Japan.

CHARLES RIVER DATA SYSTEMS

Information

FOR ADDITIONAL INFORMATION
Contact: Hank Giles
Geithner/McGowan, Inc.
(617) 875-3821

FOR IMMEDIATE RELEASE

NEW REVISION OF CHARLES RIVER'S UNOS KERNEL INCREASES SYSTEM PERFORMANCE, SUPPORTS LARGER DISKS, ENHANCES UNIX™ V COMPATIBILITY

FRAMINGHAM, Mass., Aug. 18 -- Charles River Data Systems today introduced a new revision of its UNOS operating system kernel that increases system performance and disk storage capacity, and improves compatibility with AT&T UNIX™ System V.

The new UNOS revision is available immediately and is priced at $1,000. OEM discounts are available.

The new UNOS revision supports:

- Charles River's new 3.5 MIPS (million instructions per second) VMEbus-based Motorola 16.7 MHz 68020 central processor and the 16.7 MHz 68881 floating point co-processor,
- New 540-Mbyte Winchester disk drive systems,
- 1/2-inch, 1,600, 3,200, or 6,250 bpi tape drives,
- New user and system commands and subroutines to improve compatiblity with UNIX™ System V,
- A Common Object File Format (COFF) that is similar to the COFF used in UNIX™ System V,
- The Korn Shell from AT&T's System V ToolChest™.

UNOS supports real-time applications and run-time environments. It is the real-time kernel in Charles River's UN/System V operating system, which is derived from UNIX™ System V under license from AT&T, and runs on all Charles River systems.

Charles River Data Systems was founded in 1973 as a computer peripheral supplier, and introduced the Universe 68, the first true 32-bit general purpose computer system based on a microprocessor, in September, 1981. The company also offers a family of 32-bit computer systems based on the VMEbus, and offers UniverseNet, a communications system that supports the (TOP) Technical Office Protocol local area network standards, addresses all presently defined levels of MAP 2.1 (Manufacturing Automation Protocol), and supports the International Standards Organization's Open System Interconnect standards. UniverseNet allows Universe systems to communicate with systems from other manufacturers using a non-proprietary communications protocol.

UNOS, Universe, and UniverseNet are trademarks of Charles River Data Systems. UNIX, ToolChest, and Korn are trademarks of AT&T.

CSG IMS

NEWS RELEASE

August 11, 1986

CSG IMS NOW AVAILABLE FOR OS9/68000

Clearbrook Software Group is pleased to announce the availability of their Information Management System for the OS9/68000 operating system. This version of CSG IMS is identical in features to version 1.3 on OS9/6809. CSG IMS applications are portable between OS9/6809 and OS9/68000. Data files created with CSG IMS version 1.3 on OS9/6809 are fully portable to OS9/68000. The price for CSG IMS for OS9/68000 is $495.00 (US dollars). The run-time interpreter (included with CSG IMS) is available separately for user-distrib ted applications for $100.00 (US dollars).

CSG IMS VERSION 1.3 RELEASED FOR OS9/6809

Registered users of CSG IMS (OS9/6809) may now upgrade to version 1.3. Some new features in version 1.3 include: file sorting capability, enhanced multi-user features, and enhanced terminal features. Databases created with version 1.3 of CSG IMS are fully portable to CSG IMS on OS9/68000. All CSG IMS user-written applications are fully upward-portable.

SEIKOU TO MARKET CSG IMS IN JAPAN

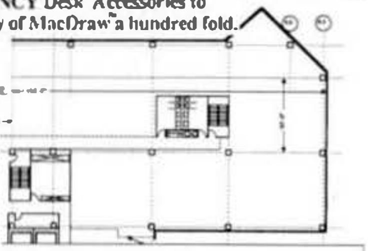
Seikou and Clearbrook Software Group have entered into an agreement to market a Japanese language version of CSG IMS. Seikou will tailor CSG IMS to work with Japanese character sets and translate the English documentation to Japanese. The Japanese version will be available exclusively from Seikou.

Clearbrook Software Group

P.O. Box 8000-469, Abbotsford, B.C. V2S 6H1
P.O. Box 8000-469, Sumas, WA 98295-8000

Turn MacDraw into a Computer-Aided-Design Workstation.....

Use the SIZER and ADJACENCY Desk Accessories to increase the accessible accuracy of MacDraw™ a hundred fold.



Use ADJACENCY to:

Move any two MacDraw objects (or groups of objects) so the nearest sides touch and the edges are coincident.

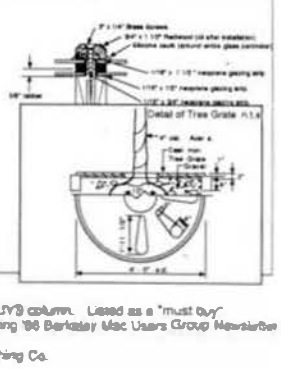
Use SIZER to:

Size selected objects (or groups) by typing in dimensions;

Report selected objects' dimensions in inches, centimeters, or in feet and inches (architectural units);

Paste reported figures on dimension lines; and

Find sums of abutting dimensions.



Features:
- Keyboard equivalents for pull-down menus.
- Accuracy to 0.0001 (inches or centimeters)
- Accuracy to 0'-0 1/64" in architectural scale.
- Accepts desktop publishing dimensions in points.
- Rapidly converts between units.
- Uses MacDraw's™ Custom Rulers to represent items over 8000 feet, inches, or cm in size.
- Control which direction the selected objects expand or contract in SIZER.
- Complete keyboard control of dialog boxes.
- Programmed for architects and engineers.
- Operates on all Macintoshes.
- Manipulates extremely small objects (overcomes the need for a zoom).
- Switcher compatible.

Named "Hot Stuff" in MACazine's July/August GOODBUYS column. Listed as a "must buy" program in the CHOICE PRODUCTS section of the Spring '86 Berkeley Mac Users Group Newsletter

"Sizer and Adjacency are impressive". Cahners Publishing Co.

To get your copy of the SIZER and ADJACENCY Desk Accessories, send a [illegible] Version 1.7 or 1.9 and $49.00 [illegible] tax) to:

JOHNSON & JOHNSON
DESIGN/BUILD
677 Grove Street
Newton, MA 02162

Telephone (617) 253-5965

[illegible]

MacDraw and Macintosh are trademarks of Apple Computer Inc.

Copyright © 1985,1986 by Eric Johnson

Classifieds As submitted - No Guarantees - As is!

DAISY WHELL PRINTERS
QUME SPRINT 9 - $900.00
QUME SPRINT 5 - $800.00
TEC FP1500 - $800.00

Winchester 10 Megabyte Drive - Seagate Model #412 - $275.00.

3 - Dual 8" drive enclosures with power supplies. New in Box - $175.00 each.

5 - Siemens 8" Disk Drives - $150.00 each.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09 CPU Card - $900.00 complete.

CDSI 20 Meg Hard Disk System w/ contoller $500.00.

Teletype Model 43 Printer - with serial (RS232) interface and full ASCII keyboard. $250.00 ready to run.

Tano Outpost II, 56K, 2-5" DSDD Drives, FLEX, MUMPS - $495.00.

MICROKEY Single Board Computer, Target 128K RAM, FORTH, FLEX, with optional 6502 CPU & ROMS -as advertised on p.51 DEC. 68' Micro Journal - $995.00.

I pay Shipping.

(615) 842-4600 M-F 9AM to 5PM EST - TOM

***

fFORTH + 3 - 5" Disks with Manual and 2 Books on FORTH - $250.00 Vakue - FREE - I pay postage.
(412) 563-5312 Jeffrey Craig, 2463 N. Meadowcroft Ave.
Pittsburg, Pa. 15216.

***

GIMIX 3 with OS9 level 2 and GIMIX 6809 CPU III board, 2-5" Disc Drives, 47 Megabyte hard disk drive, 4 I/O serial boards with 3 ports each, 9 Lear Siegler ADM22 with RS232 cables ranging from 50 to 150 ft in length, 1-512K memory board, all original disks, OS9 level 2, Basic 09, Sculptor, RMS, Sage, OS9 Micro text editor and manuals for and updates.
Make an offer.
Richard Allbert, LIFEGUARD Inc.
1-800-255-7053.

***

6805 Development Systems for MS-DOS Computers

Use the IBM PC/XT/AT or compatible MS-DOS computers to develop applications based on Motorola's 6805 single chip microcomputers. These complete integrated systems consist of a cross assembler program and one programming circuit board with menu driven driver software. The model MCPM-1 prog board supports the MC68705P3,P5,U3,U5,R3 and R5 nmos processors while the model MCPM-2 board supports the MC1468705F2 & G2 cmos versions. The programming boards connect to the computer via a serial port and allow skipping the ususal eprom programming step when used in a development environment, however, an on-board eprom programmer is provided so that the boards can be used in a stand alone mode. The system conponents are avail separately.

Complete Sys. specify MCPM-1/2 . . . . . . $495
Shipping - add 3% for U.S.A. and Canada

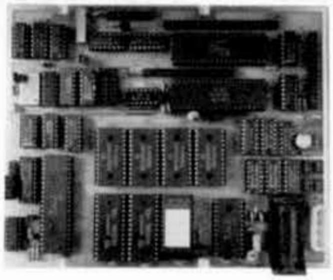
TEC
P.O.Box 53 West Glover, Vermont 05875
Phone (802) 525-3458

6809 SINGLE BOARD COMPUTER

The PT69-5 is an upgraded version of the standard PT69 board—a unique system unbeatable for its price, size, and power. It runs at twice the speed with extra features unavailable on more expensive systems.

- 2 MHZ 6809 Processor
- Floppy Disk Controller
- Port for Winchester Interface
- 4 RS-232 Serial Ports, 2 8-bit Parallel Ports
- Up to 60K RAM using 32K×8 CMOS Static
- Up to 8K Eprom (4K Standard)
- Optional Software: OX/9 or SK*DOS

$349.95



$349.95

Call or write for more details!

PERIPHERAL TECHNOLOGY
1480 Terrell Mill Rd. Suite 870
Marietta, GA 30067
404/984-0742 Telex # 880584
VISA/MASTERCARD/CHECK/COD

*OS-9 is a trademark of Microware & Motorola.

LLOYD I/O TM INC.

Lloyd I/O is a computer engineering corporation providing software and hardware products and consulting services.

19535 NE GLISAN • PORTLAND, OR 97230 (USA)
PHONE: (503) 666-1097 • TELEX: 910 380 5448 LLOYD I O

Computer Engineers

K-BASIC™ *IS HERE*

K-BASIC is a TSC XBASIC (XPC) compatible COMPILER
for OS9 & FLEX... price $199

Here at last is a compiler for BASIC that will compile all your XBASIC programs. K-BASIC compiles TSC's XBASIC and XPC programs to machine code. K-BASIC is ready now to save you money and time by teaching your computer to:
• Think Faster • Conserve Memory • Be Friendlier

Call (503) 666-1097 for our CATALOG.
We have many programs for serious software developers!

DO™

Micro BASIC for OS9... $149

A structured mirco BASIC for general system control featuring: Parameter passing, 10 string variables, 26 numeric variables, subroutines, nested loops, interactive I/O, sequential files, and time variables (for applications executing in the background required to execute procedures such as disk or file backups.) Includes the SEARCH and RESCUE UTILITIES™. (For OS9 ONLY.)

SEARCH and RESCUE UTILITIES™

for OS9... $35

A super directory search utility. Output may be piped to the included utilities to perform file: COPIES, DELETES, MOVES, LISTING (pagination), and FILTERING. Some filtering utility programs are included: of interest is the FILE DATE CHECKING utilities: YOUNGER and DRAFT (Level 2). (For OS9 Level 1 and 2.)

PATCH™

Modem Communications for OS9... $39

PATCH is a modem communications program for OS9 featuring: KEY MACROS, ASCII TEXT AND BINARY FILE UP/DOWN LOADING, PRINTER COPY, and HELP MENUS. We use it several times each day with our TELEX service. PATCH is convenient and easy to use. Key macros may be pre-stored and loaded at any time.

CRASMB™

CROSS ASSEMBLER PACKAGE
for OS9 & FLEX... all for $399
Motorola CPU's... $150
Intel CPU's... $150. Others... $150

NOW FOR OS9/68K $432

CRASMB is the highly acclaimed cross [illegible]ge for OS9 and FLEX systems. It [illegible] your [illegible] a development static[illegible] get [illegible]:

[illegible] 68[illegible] [illegible]805 6809 6811 6[illegible]02
[illegible] 8048 8051 80[illegible] [illegible] Z80
(68000 [illegible] ...$249)

CRASMB features [illegible], Condition[illegible], Long symbol names, Symbol cross reference tab[illegible] code in 4 formats (OS9, FLEX, S-1-S9, INTEL HEX).

VISA, MC, COD, CHECKS, ACCEPTED

USA:	LLOYD I/O (503 666 1097),	S.E. MEDIA (800 338 6800)
England:	Vivaway (0582 423 25),	Windrush (0692 405189)
Germany:	Zacher Computer (65 25 299),	Kell Software (06203 6741)
Australia:	Paris Radio Electronics (344 9111)	
Japan:	Microboards (0474) 22-1741	Seikou (03) 832-6000
Switzerland:	Elsoft AG (056 86 27 24)	
Sweden:	Micromaster Scandinavian AG (018 - 138595)	

K-BASIC, DO, SEARCH and RESCUE UTILITIES
PATCH, CRASMB, and CRASMB 16.32 are trademarks of LLOYD I/O
OS9 is a ™ of Microware, FLEX is a ™ of TSC

OS-9 UniFLEX
MUSTANG-020, 68020, 68881 AND MORE
HANDS-ON EXPERIENCE

The DATA-Comp Division of Computer Publishing Corporation announces their new and innovative HANDS-ON 68020 computer familiarization two day event. A chance to TRY BEFORE YOU BUY!

For two full days (Monday through Friday - excluding legal holidays) each participant will be furnished the exclusive use of a 68020 computer (MUSTANG-020). Each system will have available native C compilers, BASIC, assembler and other high level languages. Each system will be equipped with the Motorola MC 68881 math co-processor, where applicable.

Each demonstration room will contain not more than two work stations. Each system will be equipped with floppy disk, 20 megabyte winchester technology hard disk, and 2 megabyte of RAM. RAM is partitioned as 690K bytes of RAM disk and 1.2 megabyte of user RAM space.

Participants are encouraged to bring along any source level projects, for evaluation, in C, BASIC or assembler. Call for availibility of other HHLs.

Although this is not a training seminar, Data-Comp personnel are available for assistance and consultation. This event is scheduled for hands-on evaluations of the 68020 CPU, 68881 math co-processor and MUSTANG-020 system, operating in a functional environment.

Transportation to and from the airport and hotel/motel will be provided. Lunch provided both days. Chattanooga airport is serviced by American, Delta, Republic and other airlines.

COST

One person - $375.00

Two persons - $595.00

* Motel single $22.00, double $26.00
Includes satellite TV - convenient to food and shopping

DATA-COMP

A Division of
Computer Publishing, Inc.

5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

Systems available for both OS-9 and UniFLEX. Reservation should be made 15 days in advance. Attendee should initially indicate OS-9, UniFLEX or both. Special facilities available on request. Please write or call for additional information.

NOTE: Both OS-9 and UniFLEX are Unix type operating systems. Each as been enhanced in some aspect or another. Prospective attendees should have some working knowledge or experience with one of these operating systems, to gain full benefit of the session. However, a newcomer will find that it is a simple matter to be fairly proficient in using these systems in the allocated time. Special system instruction available on request. Call or write.

* Hotel/Motel cost are separate cost, not included in the basic cost shown.

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly colum has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SETC5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
* All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615) 842-4601
Telex 5106006630

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware and Motorola
*68' Micro Journal is a trademark of Computer Publishing Inc.

Videodisc
Memory Card
Optical Disc
CD-ROM
CD-I
Optical Film
LaserActive '86
Tradeshow and concurrent conferences.
Boston October 9-10-11
Insurance and Banking
Document Storage
Education
Publishing
Training
Medical

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS/9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
NEW: 68010 disassembler $100-FLEX,OS/9,UNIFLEX,OS/9-68K,MSDOS

CROSS ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS/9,UNIFLEX,MSDOS ANY 3 $100 ALL $200
specify for 180x,6502,6801,6804,6805,6809,Z8,Z80,8048,8051,8085,68000
modular cross-assemblers in C, with load/unload utilities NOW: OS/9-68K
8-bit (not 68000) sources for additional $50 each, $100 for 3, $300 for all

DE UGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS/9 $80-UNIFLEX
OBJECT-ONLY versions: EACH 50 COCO FLEX,COCO OS/9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS/9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809
6502 to 6809 $75-FLEX $85-OS/9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind. $50-FLEX $75-OS/9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $ 0-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc
non-FLEX versions include sort and resequencer only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS/ ,UNIFLEX,MS-DOS
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $12.50-SSSD/SSDD/DSDD
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNIT

CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 0207
Telephone 404-48 -4570 or 1717

We take orders at any time, but plan
long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware; COCO Tandy; MSDOS Microsoft

SOFTWARE FOR THE HARDCORE

** FORTH PROGRAMMING TOOLS from the 68XX&X **
** FORTH specialists — get the best!! **

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET COMPILATION for

6800, 6301/6801, 6809, 68000, 8080, Z80

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than complied languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH™ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

™ tFORTH and firmFORTH are trademarks of Talbot Microsystems
™ FLEX is a trademark of Technical Systems Consultants, Inc
™ CP/M-68K is trademark of Digital Research, Inc

tFORTH™
from TALBOT MICROSYSTEMS
NEW SYSTEMS FOR
6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORcisor Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

** tFORTH — exte ded fig FORTH (1 disk) $100 ($15)
with fig line editor.
** tFORTH + — more! (3 5" or 2 8" disks) $250 ($25)
adds screen editor, assembler, extended data types, utilities, games, and debugging aids.
** TRS-80 COLORFORTH — available from The Micro Works
** firm FORTH — 6809 only. $350 ($10)
For target compilations to rommable code. Automatically deletes unused code. I cludes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH +.
** FORTH PROGRAMMING AIDS — elaborate decompiler $150

** tFORTH for HX-20, in 16K roms for expansio unit or replace BASIC $170

** tFORTH/68K for CP/M-68K 8" disk system $290
Makes Model 16 a super so tware development system.

** Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) $200
— TARGET source code: 6800-$200, 6301/6801—$200
same plus HX-20 extensions— $300
6809—$300, 8080/Z80—$200, 68000—$350

Manuals available separately — price in ( ).
Add $6 system for shipping, $15 for foreign air.

TALBOT MICROSYSTEMS 1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

AVAILABLE NOW!

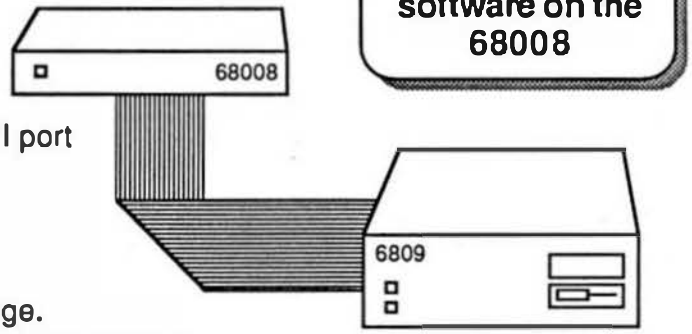
PLμS-68k (PL/9 for the 68000) running under FLEX™

- Built-in screen editor
- Built-in source-level debugger
- Byte, Integer and Long variables
- Single-pass compiler
- Direct source to object
- Compiles over 1000 lines/min
- Runs on any FLEX™ system with a spare PIA port

Develop 68000 software on your FLEX™ system. No second computer required!

The second processor module (included):

- 10MHz 68008 CPU
- 128k bytes RAM
- Case and power supply
- Plugs into Windrush UPROM-III port



Run FLEX™ software on the 68008

Other software included:

- Program loader
- 68000 FLEX™ interface package.
- Comprehensive System Monitor with source
- Hex-Binary-Hex Conversion Routines

Development is currently under way of a version for OS/9-68000. The package price includes a free copy of the OS/9 version when it becomes available.

$999
Complete

For further information, phone or write:

Worstead Laboratories
North Walsham
Norfolk NR28 9SA
England

Tel (44) 692 404086
Telex 975548 WMICRO G

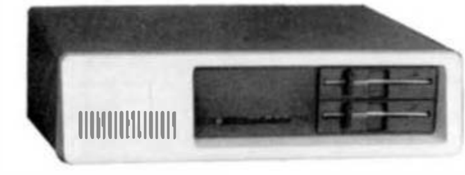
WINDRUSH
Micro Systems Ltd.

PT-69 SINGLE BOARD COMPUTER SYSTEMS

NOW WITH WINCHESTER OR FLOPPY DISK DRIVES

The proven PT-69 Single Board Computer line is expanding! Systems now can be Winchester or floppy-based. Available also in a smaller cabinet without drives for dedicated systems with no mass storage requirements.

- * 1 MHZ 6809E Processor
- * Time-of-Day Clock
- * 2 RS 232 Serial Ports (6850)
- * 56K RAM 2K/4K EPROM
- * 2 8-bit Parallel Ports (6821)
- * 2797 Floppy Disk Controller



Winchester System

Custom Design Inquiries Welcome

Floppy System

*PT69XT WINCHESTER SYSTEM $1795.95
Includes 5 MEG Winchester Drive, 2 40 - track DS/DD Drives, Parallel Printer Interface + choice of OS/9 or STAR-DOS.

*PT69S2 FLOPPY SYSTEMS $895.95
Includes PT69 Board, 2 DS/DD 40 - TRK 5 1/4" drives, cabinet, switching power supply, OS/9 or STAR-DOS.

*PT-69A ASSEMBLED & TESTED BOARD	$879.00
*OS/9	$200.00
*STAR-DOS	$ 50.00

PERIPHERAL TECHNOLOGY
1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067
Telex #880584

VISA/MASTERCARD/CHECK/COD

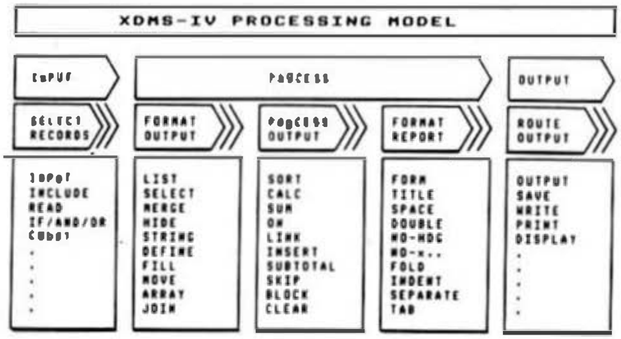
☎ 404/984-0742

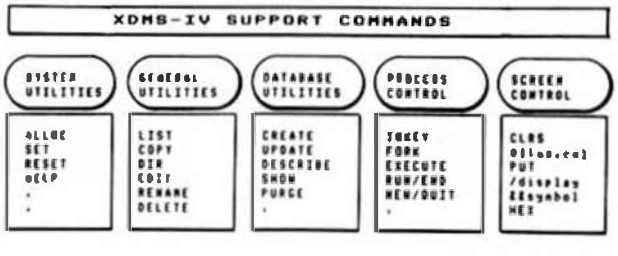
CALL OR WRITE FOR ADDITIONAL CONFIGURATIONS

™OS9 is a trademark of Microware

XDMS-IV

Data Management System





Up to 32 groups/fields per record! Up to 12 character field names! Up to 1024 byte records! Input-Process-Output (IPO) command structure! Upper/Lower case commands! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a 'process' file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...). The possibilities are unlimited...

XDMS-IV for 6809 FLEX, STAR-DOS, SKsDOS (9* or 6*t......$350.00+P&H
Order by Phone: 615-842-4600/4601 - (VISA and MasterCard accepted)
Or write: South East Media, 5900 Cassandra Smith, Hixson, Tenn 37343

WESTCHESTER Applied Business Systems
2 Pea Pond Lane, Briarcliff Manor, N.Y. 10510 Tel 914-941-3552(Eves)
(FLEX(tm) Technical Systems Consultants, SK*DOS(tm) STAR-KITS Corp.

BOARDS & PARTS FOR GIMIX SYSTEMS

CONTACT GIMIX FOR PRICES, DETAILS, AND REQUIRMENTS FOR HARD DISK AND OTHER MASS STORAGE UPGRADES.

THE GIMIX CLASSY CHASSIS #19 consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board $1490.19

#22 Triple Disk Regulator Card			$88.22
#93 Baud Rate Generator Board			$88.93
#23 Missing Cycle Detector			$30.23
#92 Filler Plate	$14.92	50 Hz Option	$30.00
Cable sets 8" with Back Panel connector			$29.25
for two 8" external drives	$44.26	for two 5" drives	$34.96

CPU BOARDS

#01 GMX III CPU & OS-9 GMX III			$1890.01
#02 GMX III CPU & UniFLEX III			$1990.02
The #05 GIMIX 6809 PLUS CPU Board			$578.05
Options: GIMIX DAT	$35.00	9511A	$312.00
SWIP Dat	$15.00	9512	$265.00
#03 6800 CPU			$224.03
#06 6800 CPU w/Timers			$288.06
6800 Baud Rate Option			Add $30.00

FLOPPY DISK CONTROLLER

#68 DMA	$588.68

MEMORY BOARDS FOR 6809/68020 SYSTEMS

#72 256KB CMOS STATIC RAM board with battery back up (specify system)	$648.72

MEMORY BOARDS (6800/6809 SYSTEMS ONLY)

#34 8K PROM Card	$98.34
#32 16 Socket PROM/ROM/RAM Board, 24 pin	$238.32
#31 16 Socket Universal Memory Board, 24/28 pin	$268.31

INTELLIGENT I/O PROCESSOR BOARDS

significantly reduce systems overhead by handling routine I/O functions, freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2k baud. For use with GMX III and 020 systems.

#11 3 Port Serial-30 Pin (OS9)	$498.11
#14 3 Port Serial-30 Pin (UniFLEX)	$498.14
#12 Parallel-50 Pin (UniFLEX-020)	$538.12
#13 4 Port Serial-50 Pin (OS9 & UniFLEX-020)	$618.13
#15 24K Version of #11, with either large input or output buffers (specify)	$648.15

I/O BOARDS (6800/6809 SYSTEMS ONLY)

#41 Serial, 1 Port	$88.41
#43 Serial, 2 Port	$128.43
#46 Serial, 8 Port (OS9/FLEX only)	$318.46
#42 Parallel, 2 Port	$88.42
#44 Parallel, 2 Port (Centronics pinout)	$128.44
#45 Parallel, 8 Port (OS9/FLEX only)	$198.45
#50 I/O for RS-232C, 423, 422-w/6850	$244.50
#52 SSDA with 6852	$254.52
#54 ADLC with 6854	$268.54

CABLES FOR I/O BOARDS — SPECIFY BOARD

#95 Cable sets (1 needed per port)	$24.95
#51 Cent. B.P. Cable for #12 & #44	$34.51
#53 Cent. Cable Set	$36.53

OTHER BOARDS & PARTS

#66 Prototyping Board-50 Pin	$56.66
#33 Prototyping Board-30 Pin	$38.33
Windrush EPROM Programmer S30 (OS9/FLEX 6809 only)	$545.00
#76 Video Board-80 x 24	$398.76
#08 Relay Driver Package	$1128.08
#86 Above without Relays	$538.86
Opto Board	$348.85
Binder, 3"	$12.00
Binder, 2"	$9.00

8" DRIVE CABINET & PARTS

2 8" DS00 Drives, Cabinet & Cables 60 Hz only	$1698.88
Cabinet Only for 8" Drive	$848.10
220v/50 Hz Option Add	$30.00
Cable Set-Internal for 2 Drives	$44.02
Cable Set-Internal for 4 Drives	$67.84
Cable from 8" Cab. to Mainframe	$45.81
8" Filler Plate	$14.83

SOFTWARE:

GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #68 controller. When ordered with controller, FLEX is $30.00

GIMIX versions of FLEX	$90.00
GMX VDisk for FLEX 09	$100.00
GMXBUG: PROMs & Manual	$148.65
Boot or Video/Boot PROMS (6809)	$30.00
GIMIX Boot PROM for UNIFLEX	$50.00
RMS (OS9)	$250.00
DO (OS9)	$70.00
OS-9 GMX III Update w/CPU SPPTROM	$125.00
I/O PROMS w/Update	$40.00
GMXBUG/FLEX/VDISK w/OS-9 III update	Add $175.00
RAM Disk for OS-9	$125.00
O-FLEX	$250.00
OS9 GMX I	$250.00
OS9 GMX II	$500.00
SCULPTOR-6809 (UniFLEX/OS-9)	$995.00
SCULPTOR—68020 (UniFLEX)	$1595.00

CONTACT GIMIX FOR PRICES AND AVAILABILITY OF OPTIONAL UNIFLEX AND OS9 LANGUAGES AND OTHER SOFTWARE.

ALL PRICES ARE F.O.B. CHICAGO.

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

LIMITED WARRANTY

GMX INC. ("GMX") Warrants its products against defects in material and workmanship for a period of ninety days from the date of shipment. The obligation of GMX is limited to the repair or replacement of any product, free of all charges, which proves defective during this period. This warranty does not cover damage due to accidents, negligence, abuse or tampering.

GMX MAKES NO OTHER WARRANTIES OR GUARANTEES, EXPRESS, STATUTORY, OR IMPLIED, OF ANY KIND WHATSOEVER WITH RESPECT TO ANY PRODUCT PURCHASED, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS HEREBY DISCLAIMED BY GMX AND EXCLUDED FROM ANY AGREEMENT MADE BY GMX.

GMX will not be responsibile for any damage of any kind

not covered by the exclusive remedies set forth in this limited warranty. GMX will not be responsible for any special, indirect, or consequential damage caused by its products.

GMX products are not for consumer use. GMX expressly disclaims all warranties on any of its products which may be included in any product normally used for personal or family purposes.

Contact GMX by mail at 1337 West 37th Place, Chicago, IL 60609; or phone at (312) 927-5510; if your product is defective to arrange for its repair or replacement under this warranty.

Repair charges for GIMIX products after warranty period will be $35.00 per hour per board (minimum $35.00) plus parts. Customer pays freight charges both ways. If GIMIX determines that replacement is desirable instead, we will notify you. Charges for checking out complete system will be $500.00 plus parts, freight, and necessary board repairs.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

GIMIX inc.
1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Introducing ....

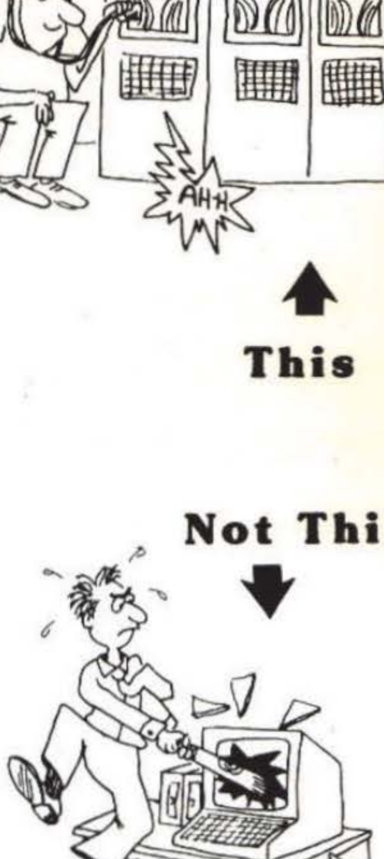
S - 50 BUS / 68XX

Board and/or Computer
Terminals-CRTs-Printers
Disk Drives-etc.

REPAIRS

This

Not This



NOW AVAILABLE TO ALL S50/68XX USERS

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to 'ALL' S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SSB - HELIX and others, including the single board computers. *Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

1. If you require service, the first thing you need to do is call the number below and describe your problem and *confirm a Data-Comp service & shipping number*! This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

2. All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

3. If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates *must be requested*. Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with the item to be serviced. Please note that estimates are given in dollar amounts only.

4. Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

☎
(615)842-4607
Telex 5106006630